Los Ficheros en el AMSTRAD PCW 8256/8512 y MALLARD BASIC para CPCs.

VICTOR J. CAMPO

ra-ma

Los Ficheros en el AMSTRAD PCW 8256/8512 y Mallard Basic para CPC 6128

Víctor J. Campo
Licenciado en Ciencias Económicas

ra-ma

Los Ficheros en Amstrad PCW-8256/8512
y Mallard Basic para CPC-6128

(c) Víctor J. Campo

(c) Copyright de la edición RA-MA 1987

Reservados todos los derechos

Ninguna parte de este libro puede ser reproducida, grabada en sistema de almacenamiento o transmitida en forma alguna ni por cualquier procedimiento, ya sea electrónico, mecánico, reprográfico, magnético o cualquier otro, sin autorización previa y por escrito del editor.

Editado por:

RA-MA Editorial
Ctra. de Canillas, 144
28043 MADRID
Telefs. (91) 200 97 46/47

ISBN - 84-86381-30-4
Depósito Legal M. 28970 - 1987
Fotocomposición: Comprint, S.A.
Impreso en España
Signo Impresores, S A Madrid

Dedicatoria:
A Ricardo y María Luisa.

Introducción

Son varias las razones que me han animado a escribir estas páginas. La primera es que los modelos PCW de Amstrad son mucho más que una herramienta para tratamiento de textos. El lenguaje BASIC que incorporan se muestra muy potente sobre todo para el trabajo de ficheros, sin embargo la bibliografía existente incide más en el trabajo de Locoscript y descripción del sistema operativo CP/M 3.0 Plus, que en este campo de la programación.

La segunda razón, es la de intentar, de alguna forma, la compatibilidad entre estos modelos y el CPC 6128, que dispone también de la misma versión del S.O. CP/M y cuyo lenguaje BASIC residente en ROM, aún habiendo sido creado por la misma firma, Locomotive Software, no permite un fácil intercambio de programas, que enriquecería sin duda a ambos modelos. El capítulo 0 está dedicado a mostrar como puede el CPC 6128 trabajar en Mallard BASIC y por tanto incorporar los programas del PCW a su biblioteca.

De esta forma, el libro está dedicado tanto a los usuarios de los PCW, como a los del CPC 6128, que se sienten animados a trabajar con su ordenador en Mallard BASIC.

El contenido del libro está dividido en 3 partes. La primera, capítulos 0 y 1, trata de la instalación de Mallard BASIC en los Amstrad PCW y CPC 6128. La segunda que comprende los capítulos 2 y 3, está dedicada a profundizar en el manejo de matrices (variables indexadas), como estructuras de almacenamiento transitorio y manipulación de datos y a la ordenación de datos, con descripción teórica y práctica de diversos métodos, incluída la ordenación por fechas, para el trabajo de gestión. Por último, los capítulos dedicados a los ficheros: secuenciales, aleatorios y de acceso por claves.

Todos los capítulos se ilustran con numerosos ejemplos y programas útiles, sobre los temas tratados, además se incluyen como apéndices a los mismos programas completos, que además de su valor didáctico, permiten el uso directo como programas de aplicación.

El Autor

Indice de capítulos

CAPITULO 0. Mallard BASIC para CPC 6128. Instalación 1
1. Mallard BASIC para PWC 8256 y 8512. Instalación 11
2. Matrices 19
3. Ordenación 61
4. Introducción a los ficheros 83
5. Ficheros secuenciales 89
6. Ficheros de acceso directo 113
7. Ficheros de acceso por claves (Jetsam) 133
8. Ficheros creados por CP/M 159

APENDICES 1. Código ASCII 173
2. La impresora de los PCW 177
3. Códigos de error de Mallard BASIC 185
4. Conversión de números. Cambio de base 187
5. Conversión de ficheros locoscript a ASCII 189
6. Códigos de control 191
7. Editor sencillo para ficheros de órdenes 195

Relación de programas y ficheros contenidos en el libro

CAPITULO 0

1. Fichero instalación Mallard BASIC en los CPC Amstrad
2. Fichero definición teclado para los CPC Amstrad

CAPITULO 1

3. Fichero instalación Mallard BASIC en los PCW Amstrad
4. Fichero definición teclado para los PCW Amstrad

CAPITULO 2

5. Matrices. Carga de datos
6. Matrices. Carga de datos
7. Suma de elementos de una tabla
8. Suma total de elementos de una tabla
9. Suma por filas
10. Suma por columnas
11. Búsqueda del mayor o menor de una lista numérica
12. Búsqueda del mayor o menor de una lista alfanumérica
13. Rutinas de corrección de datos de una matriz
14. Rutina de eliminación de datos numéricos de una lista
15. Rutina de eliminación de datos alfanuméricos de una lista
16. Rutina de agregación de datos a una lista. Inserción por la derecha
17. Rutina de agregación de datos a una lista. Inserción por la izquierda
18. Rutinas de búsqueda por igualdad
19. Rutinas de búsqueda por inclusión
20. Búsqueda en listas alfanuméricas ordenadas
21. * Notas medias
22. * Producto de una matriz por un escalar
23. * Inversa de una matriz
24. * Determinante de una matriz cuadrada
25. * Resolución de sistemas de ecuaciones
26. Ordenación por intercambio
27. Ordenación por el método de la burbuja
28. Ordenación por el método de la burbuja con marca

29. Ordenación por el método del intercambio retardado
30. Ordenación por el método de intercambio retardado con doble pasada
31. Ordenación por el método de SHELL-METZNER
32. Ordenación por el método de Quickshort
33. Rutinas para el trabajo de ordenación. Entrada de datos
Grabar en fichero
Leer fichero
Salida de datos
33. * Programa general de ordenación aplicable a cualquier rutina de las descritas
34. * Programa de ordenación de fechas

CAPITULO 5

35. Rutinas de lectura de ficheros secuenciales con FOR
36. Rutinas de lectura de ficheros secuenciales con bucle contador
37. Rutinas de lectura de ficheros secuenciales con bucles WHILE
38. Rutinas de lectura de ficheros de un solo dato
39. Grabación de datos. Matriz-Fichero
40. * Pgm. de gestión de biblioteca
41. * Juego educativo de preguntas y respuestas. Generador de preguntas y juego

CAPITULO 6

42. * Agenda de direcciones
43. * Gestión de videoclubs

CAPITULO 7

44. * Base de datos. Proveedores
45. * Gestión de biblioteca (versión CPC)
46. * Gestión de biblioteca (versión PCW)

APENDICES

47. Rutina definición códigos impresora (PCW)
48. Rutina control de errores
49. Conversión de números. Cambio de base
50. Conversión de ficheros LOCOSCRIPT a ASCII
51. Editor sencillo de textos en Mallard BASIC

NOTA * Programas de utilidad completos

NOTA Con el fin de facilitar el trabajo, se han incluido los programas y rutinas en disco, listos para su uso Existen dos versiones una para los CPC y otra para PCW, para su adquisición dirigirse a

Editorial RA MA
Carretera de Canillas, 144
28043 MADRID
Tel 2 00 97 46/47

CAPITULO 0

MALLARD BASIC PARA CPC 6128

1. INTRODUCCION

2. COMPATIBILIDADES DE LOS LENGUAJES

3. MALLARD BASIC PARA CPC 6128

0. Mallard BASIC para CPC 6128

1. INTRODUCCION

Son dos los objetivos de este capítulo, por un lado el análisis de la posible compatibilidad entre el LOCOMOTIVE BASIC de los CPC y en particular del CPC 6128 con el MALLARD BASIC de los PCW y por otro la posibilidad de utilizar MALLARD BASIC en el CPC 6128.

2. COMPATIBILIDADES DE LOS LENGUAJES

Analizando el grado de similitud de algunas órdenes en ambas versiones, desde el punto de vista de un CPC 6128 y como consecuencia la posibilidad de adaptación de programas de una a otra versión BASIC, se desprende que existe un alto grado de dificultad en dicho intercambio, tanto mayor, cuanto más alto es el nivel de programación.

A título de información se incluyen a continuación algunas de las órdenes y funciones del BASIC del CPC 6128 y su equivalente al Mallard del PCW.

— ON BREAK CONT - OPTION RUN (activa) OPTION STOP (desactiva).

— CALL - CALL la dirección está contenida en una variable en vez de utilizarse el número directamente.

— CHAIN, MERGE, CHAIN MERGE. Funciona de forma análoga pero Mallard BASIC puede conservar el valor de las variables.

— Funciones para trabajar con números INT, CINT, CREAL, FIX. Existen también en Mallard, e incluso se duplican debido a la existencia de numeros reales en precisión simple y en doble precisión (ver capítulo 2) por ejemplo CREAL da lugar a CDBL y CSNG (doble y simple respectivamente).

— Catálogo. El CPC dispone de CAT en BASIC y DIR en CP/M, en Mallard se dispone de DIR, FILES que necesita una plantilla de búsqueda y FIND$ para buscar un fichero concreto.

— CLEAR. En su uso más simple es igual en ambos BASIC.

— CLS. No está disponible «desgraciamente» en el Mallard y debe sustituirse por PRINT CHR$ (27) + "E" + CHR$ (27) + "H" que borra y lleva el cursor al

origen o bien solo la mitad de la sentencia que no mueve el cursor, pero si borra. (Ver apéndice 6).

— Medida de los ángulos. Mientras que el CPC permite trabajar en grados sexagesimales y en radianes (DEG y RAD), el Mallard sólo lo hace en radianes. Esta dificultad no es mucha si se tiene en cuenta la conversión de grados a radianes ($360^\circ = 2\pi$ radianes)

$$\text{Rad.} = \frac{G^\circ \cdot \pi}{180} \quad \text{siendo } G = N^\circ \text{ grados sexagesimales a convertir}$$
$$\text{Rad.} = \text{Radianes equivalentes}$$

La función PI no existe en Mallard y por tanto debe ser sustituída por su valor.

— Funciones de error. ERL y ERR son iguales en ambas versiones y la DERR equivale en Mallard a OSERR.

— Fin de fichero. En ambas versiones se utiliza EOF, aunque en Mallard debe llevar además el indicativo de número de fichero, puesto que pueden existir varios abiertos simultáneamente.

— Definir teclado. Mallard no dispone de KEY ni KEY DEF, por tanto la redefinición del teclado ha de hacerse con CP/M (SET KEYS)

— Impresora. Las órdenes LIST y PRINT para impresora se utilizan añadiendo el indicador de canal # 8, en Mallard son equivalentes a L LIST y L PRINT respectivamente.

— Pantalla. Frente a los tres modos de pantalla del CPC, el PCW sólo dispone de uno, de 90 columnas × 32 líneas, sin colores.

— Sonido. No existen en Mallard sonidos, a excepción del pitido emitido en ambos modelos por PRINT CHR$ (7).

— INKEY$. Se utiliza de igual forma en ambos modelos. Mallard no dispone de INKEY.

— LOCATE. Debe sustituirse en Mallard por la expresión.

```
PRINT CHR$ (27) + "Y" + CHR$ (32 + F) + CHR$ (32 + C)
```

donde F y C son respectivamente fila y columna.

— Ficheros. Es aquí donde la diferencia es mayor, ya que el CPC 6128 sólo dispone de ficheros secuenciales, mientras en Mallard se dispone de secuenciales, aleatorios y por claves. (El CPC dispone de ficheros aleatorios a través de extensiones RSX creadas por RANDOM.BIN).

3. MALLARD BASIC PARA CPC 6128

La posibilidad de utilizar Mallard BASIC en un CPC 6128 es mucho más interesante que la de tratar de traducir programas de una versión a otra.

Quizá algunos lectores piensen que no es posible utilizar el Mallard BASIC del PCW por el CPC 6128, pero esta posibilidad existe y es sencilla de realizar.

Si ha observado el contenido del disco del PCW verá que el BASIC está contenido como fichero CP/M ejecutable (BASIC. COM) y que ambos modelos utilizan la misma versión de sistema operativo, CPM 3.0 Plus.

La única dificultad es que el formato de disco utilizado por ambos es distinto, pero el PCW es muy receptivo con los discos del CPC 6128.
Veamos el proceso paso a paso de formación de un disco para CPC 6128 que contenga el Mallard BASIC del PCW.

1) Con el CPC 6128 y en un disco virgen formateado (system format) mediante DISCKIT 3 (cara 1 discos de sistema), hagamos una copia de los ficheros SUBMIT.COM, SETKEYS.COM y C10CPM3.EMS. Esta copia puede realizarse con DISCKIT 3, borrando posteriormente lo que no interese o bien mediante PIP.COM.

Para aquellos usuarios no familiarizados con PIP, se indica a continuación, cómo pueden copiar los archivos mencionados con dicha orden.

— Entrar en CP/M mediante CP/M.

— Tras la impronta A > teclear A > PIP y aparecerá una nueva impronta, consistente en un asterisco, añadir ahora la expresión

* B: = A: SUBMIT.COM

— Cuando la pantalla lo indique, sacar el disco de sistema e insertar el virgen para grabar el fichero leído anteriormente. Tenga en cuenta que la unidad A: será el disco fuente y la B: el disco destino.

— Una vez copiado el fichero, repetir para los otros dos indicados anteriormente.

2) Aunque no es estrictamente necesario, copie también del disco de sistema ED.COM y DIR.COM, le será de utilidad tenerlos en el mismo disco. Utilice PIP en la forma indicada.

3) Lleve el disco tal como lo ha preparado a un PCW, instale el CP/M en el PCW, llame al programa PIP tal como lo hizo en su CPC, una vez obtengan la impronta *, teclee

* B: = A: BASIC.COM

Cuando se lo indique saque el disco fuente del PCW e introduzca su disco preparado en el CPC, de esta forma habrá copiado el pgm BASIC.COM del PCW en un disco de CPC (el disco A: será el maestro del PCW y el B: el preparado por el CPC).

Ahora ya tenemos el Mallard BASIC preparado para ser usado por el CPC 6128, pero deben añadirse dos ficheros más, el primero que llamaremos TECLADO.KEY que nos va a permitir redefinir el teclado, para hacerlo más operativo y el segundo, al que daremos por nombre PROFILE.SUB que permitirá el autolanzamiento del

BASIC, con la orden única CP/M cuando se desee cargar el Mallard BASIC en el ordenador CPC 6128.

4) Para crear el fichero TECLADO.KEY cuyo contenido será

```
E 128 "RUN^M"
15N" '128'"
E 129 "LIST "
13N"^'129'"
E 130 "?CHR$(27)+`"E "+CHR$(27)+^"H^"^M"
14N"^'130'"
E 131 "DIR^M"
5N"^'131'"
E 132 "FOR "
20N"^'132'"
E 133 "NEXT"
12N"^'133'"
E 134 "INPUT "
4N"^'134'"
E 135 "EDIT "
10N" '135'"
E 136 "CHR$("
11N"^'136'"
E 137 "GOTO "
3N"`'137'"
8N"`A"
1N"`F"
```

(los símbolos que aparecen en el listado son ^ = ↑ de la tecla £, ' = apóstrofe de la tecla 7, del teclado numérico superior).

Puede utilizarse el programa ED.COM, que es el editor de CP/M y que grabamos anteriormente (puede usar otro editor o el incluido en el apéndice 7).

Si no conoce el funcionamiento de este editor siga atentamente las instrucciones que se indican a continuación, ya que el manejo del editor de CP/M no es precisamente sencillo y la disposición de todas sus funciones escapan de los objetivos de este libro.

Entre en CP/M y tras la impronta A > invoque al editor mediante la orden

A > ED TECLADO.KEY

que además de llamar al editor crea un fichero de nombre TECLADO.KEY. En pantalla aparecerá el mensaje NEW FILE indicando que ese fichero no existía en el disco.

Como verá aparece también una nueva impronta :* teclee a continuación una i y presione RETURN, en pantalla habrá ido obteniendo (i indica a ED que se van a insertar textos en el fichero)

```
A > ED TECLADO.KEY
NEW FILE
      1 :  *i
      1 :
```

La impronta 1: le indica que se encuentra en la primera línea del fichero y que puede introducir datos, introduzca E 130 "RUN ↑ M" y presione RETURN, el editor pasará a la segunda línea y así sucesivamente.

Teclee todas las líneas indicadas anteriormente, con cuidado de no cometer errores, los cursores no funcionarán, si tiene que borrar caracteres, utilice CONTROL + H (^H).

Cuando haya terminado, presione CONTROL + Z (^Z) para salir de la inserción. Nuevamente aparecerá la impronta :* teclee ahora e seguido de RETURN

```
:* e
```

y el fichero será grabado en el disco.

Puede comprobar su contenido con la orden TYPE de CP/M. Si tiene dificultades, borre el fichero e inténtelo de nuevo. Por si le son de ayuda, se indican algunas funciones de ED

# a	pasa las líneas del fichero del disco al buffer del editor (carga del fichero)
# t	Lista el fichero
n : k	Borra la línea n
i	Inserta
^z	Sale de la inserción
^c	Abandona ED
e	Graba el fichero

Para modificar una línea es preferible borrarla e insertar una nueva. En todo caso y si desea mayor conocimiento del funcionamiento de ED consulte el capítulo 8 o un manual dedicado a CP/M.

5) Si consiguió realizar el fichero anterior, no tendrá ninguna dificultad en crear con ED el siguiente, de nombre PROFILE.SUB y cuyo contenido es

```
TECLADO.KEY
BASIC INICIAR
```

El proceso visto en pantalla sería

```
A > PROFILE.SUB _ _ _ _ _ _ _ _ _ _ PROFILE SUB + RETURN
NEW FILE
      : *i _ _ _ _ _ _ _ _ _ _ _ _ i + RETURN
    1 : TECLADO.KEY_ _ _ _ _ _ _ TECLADO KEY + RETURN
    2 : BASIC INICIAR _ _ _ _ _ _ BASIC INICIAR + RETURN
    3 :_ _ _ _ _ _ _ _ _ _ _ _ _ _^z
      : *e_ _ _ _ _ _ _ _ _ _ _ _ e + RETURN
A >
```

6) Como última operación vamos a crear un programa BASIC con el Mallard BASIC copiado anteriormente, para poder trabajar con él, entre en CP/M y posteriormente teclee

A > BASIC

y aparecerá el mensaje de presentación del Mallard

Mallard - 80 BASIC with Jetsam ....

indicándole el espacio libre para BASIC. Tras el mensaje OK equivalente al Ready del BASIC del CPC 6128, teclee este sencillo programa

```
10 WIDTH 79
20 NEW
```

grábelo mediante

SAVE "INICIAR"

Esto permite adaptar el número de columnas, de 90 en que trabaja el PCW a las 80 del modo 2 del CPC.

Como comprobación final, use DIR para ver el contenido del disco, recuerde que si está en Mallard, esta orden es válida. El contenido de su disco debe ser el siguiente (si copió DIR.COM puede utilizar la orden DIR [FULL] que da mayor información).

```
dir
A  SUBMIT   COM   SETKEYS  COM   BASIC    COM   ED       COM
A  C10CPM3  EMS   DIR      COM   INICIAR  BAS   PROFILE  SUB
A  TECLADO  KEY
A>dir [full]

Scanning Directory

Sorting  Directory

Directory For Drive A   User  0
```

```
    Name      Bytes   Recs   Attributes       Name      Bytes   Recs   Attributes
----------- ------ ------ ------------ ----------- ------ ------ ------------
BASIC    COM    28k    224 Dir RW       C10CPM3  EMS    25k    200 Dir RW
DIR      COM    15k    114 Dir RW       ED       COM    10k     73 Dir RW
INICIAR  BAS     1k      1 Dir RW       PROFILE  SUB     1k      1 Dir RW
SETKEYS  COM     2k     16 Dir RW       SUBMIT   COM     6k     42 Dir RW
TECLADO  KEY     1k      3 Dir RW

Total Bytes     =     89k  Total Records =     674  Files Found =    9
Total 1k Blocks =     89   Used/Max Dir Entries For Drive A:   11/  64
```

El primer directorio corresponde a la orden DIR desde Mallard BASIC o bajo CP/M como orden permanente y el segundo a DIR.COM de CP/M (orden transitoria).

Desde este momento para disponer de Mallard BASIC en su CPC 6128, basta con que introduzca el disco formado y llame a CP/M mediante

CP/M

Las teclas de borrado CLR y DEL funcionarán normalmente, así como los cursores izquierdo y derecho, pero no los cursores arriba y abajo.

Las teclas de función contienen los siguientes comandos BASIC

f0 RUN + RETURN
f1 LIST
f2 Borrar pantalla (CLS)
f3 DIR + RETURN
f4 FOR
f5 NEXT
f6 INPUT
f7 EDIT
f8 CHR$(
f9 GOTO

Estos comandos se consiguieron con el fichero TECLADO.KEY, si lo desea, puede modificarlo, para introducir otro más acorde con sus gustos y hábitos de programación, basta con cambiar en las líneas que comienzan por E, el contenido entrecomillado.

CAPITULO 1

MALLARD BASIC PARA PCW 8256 Y 8512. INSTALACION

1. GENERALIDADES

2. INSTALACION

3. CONSECUENCIAS DE LA INSTALACION

1. Mallard BASIC para PCW 8256 y 8512. Instalación

1. GENERALIDADES

La instalación de Mallard BASIC en su ordenador es inmediata, basta con cargar CP/M desde la cara 2 de los discos de sistema y teclear BASIC tras la impronta A >, de esta forma el programa BASIC.COM de CP/M es ejecutado y puede disponerse a trabajar con la versión BASIC de los PCW de Amstrad.

Dos son los inconvenientes de este sistema, por un lado las órdenes de Mallard BASIC, habrán de ser deletreadas, lo cual produce un cierto grado de lentitud en aquellas personas con poca práctica de mecanografía, a la vez que provoca frecuentes errores de síntaxis. Por otro lado resultaría más cómodo que con el simple hecho de encender el ordenador, el BASIC estuviese disponible, tal como lo hace LocoScript.

Vamos a tratar de resolver en este capítulo ambos inconvenientes y además simultáneamente, es decir obtener una instalación con autoarranque y un teclado especialmente preparado para programar en BASIC, tal como se hizo anteriormente para los CPC.

2. INSTALACION

Para la preparación de un disco "llave en mano", que es como se conoce a las instalaciones de aplicaciones automatizadas, es necesario preparar convenientemente un disco, incluyendo en él los programas CP/M que permiten dicha operación.

Proceda de la siguiente forma:

1) Formatear un disco virgen en formato de sistema. Puede utilizar para ello el programa DISCKIT de CP/M.

2) Mediante PIP copie en el disco los siguientes programas.

— J14SCPM3.EMS. Si no dispone en su juego de discos de este programa en concreto, puede utilizar otro cuyo nombre difiera en los 3 caracteres posteriores a la J, pues se tratará de otra versión CP/M pero válida a todos los efectos.

Este programa se encarga de iniciar automáticamente el CP/M, en el momento del encendido. Cualquier disco conteniendo este programa se convierte en "Disco de arranque diario CP/M".

— SUBMIT.COM. Este programa es necesario para manejar el fichero PROFILE.SUB que será creado más tarde y almacenado en el mismo disco.

— SETKEYS.COM. Igualmente es necesario para manejar el fichero TECLAS.KEY que se describe a continuación y que guardará la información relativa a la definición del teclado, adaptándolo al BASIC.

— ED.COM. Grábelo en su disco, será utilizado únicamente en la creación de los dos ficheros mencionados y puede posteriormente borrarse a fin de liberar espacio en el disco.

— LANGUAJE.COM. Este programa sólo debe grabarlo si desea modificar el lenguaje inicial de su ordenador, que si es castellano corresponde a LANGUAGE 7 (ver fichero PROFILE.SUB).

La forma de realizar las copias no es única, puede copiarse el disco entero (cara 2) con DISCKIT y borrar con ERA posteriormente los programas y ficheros que no interesen o bien utilizar la orden PIP en la siguiente forma.

Entre en CPIM y tras la impronta A > teclee

A > PIP

obtendrá una nueva impronta, formada por el símbolo * (asterisco) teclee ahora el primer programa a copiar

* B: = A: J14SCPM3.EMS

el disco identificado como B: es el virgen y el A: el de sistema (cara 2). En unidades con un solo disco (PCW 8256) debe procederse al cambio de disco, cuando se indique por pantalla.

Repita la operación para SUBMIT.COM, SETKEYS.COM y ED.COM, también para LANGUAGE.COM si opta por cambiar de lenguaje (en principio puede no hacerlo y posteriormente incluirlo si lo desea).

3) Una vez copiados los programas vamos a proceder a confeccionar un archivo, al que denominaremos TECLAS.KEY y que contendrá información sobre la adaptación del teclado a BASIC, de forma que con una sola pulsación de tecla, se obtengan órdenes completas.

Puede utilizarse en este fin cualquier tratamiento de textos, que genere textos ASCII, pero no LocoScript directamente(1), si no dispone un tratamiento de textos apropiado, utilice el editor de textos de CP/M ED.COM, que copió anteriormente. El manejo de este editor no es precisamente sencillo, ni cómodo, pero dado que el

(1) LocoScript no genera ficheros ASCII, pero en el apéndice 5, se indica como transformar ficheros en este formato a ASCII, de forma que puede, siguiendo los pasos oportunos, utilizar LocoScript a los fines de este capítulo, además el apéndice 7 le muestra un pequeño editor de fácil manejo apropiado a estos fines. También disponen los PCW del programa RPED.BAS que es un editor escrito en BASIC, cualquiera de las opciones expuestas puede producir el fichero indicado.

fichero es pequeño, no le resultará difícil. En el capítulo 0 y 8 puede obtener información sobre como se maneja. Dos recomendaciones al efecto: es preferible borrar una línea (k) mal escrita, que intentar rectificarla, si la línea incorrecta aún no ha sido introducida (RETURN) puede retroceder sobre lo escrito borrando, mediante ^H (ALT + H).

Cuando invoque a ED para crear el fichero, cuyo contenido línea a línea, se incluye a continuación, utilice la orden

A > ED TECLAS.KEY

y que corresponde al nombre adoptado para este fichero.

```
E 129 "LIST "
E 130 "SAVE^""
E 131 "LOAD^""
E 132 "CHR$("
E 133 "RUN^M"
E 134 "AUTO"
E 135 "EDIT "
E 136 "PRINT CLS$^M"
E 151 "PRINT CLS$"
E 152 "ELSE "
E 153 "FOR "
E 154 "GOTO "
E 155 "INPUT "
E 156 "PRINT FN LOCATE$("
E 157 "NEXT "
E 158 "THEN "
02 "^'129'"
02 S "^'130'"
00 "^'131'"
00 S "^'132'"
73 "^'133'"
77 S "^'134'"
62 A SA "^'151'"
58 A SA "^'152'"
53 A SA "^'153'"
52 A SA "^'154'"
35 A SA "^'155'"
36 A SA "^'156'"
46 A SA "^'157'"
51 A SA "^'158'"
```

NOTA El símbolo ^ del listado corresponde a

Si las palabras BASIC utilizadas no le son de utilidad y prefiere otras, puede modificarlas creando un fichero más a su gusto.

En el capítulo 0 se describe sucintamente la forma de crear este tipo de ficheros con el editor ED de CPIM.
4) Una vez corregido y grabado el anterior fichero, hay que confeccionar otro, con ED o con el tratamiento de textos, si optó por esta solución.

El contenido de este fichero es muy simple

setkeys teclas.key
basic iniciar

El nombre «obligatorio» será PROFILE.SUB.

5) Sólo queda una última cosa, definir el programa "INICIAR" incluido en el anterior fichero.

Se trata de un programa BASIC muy corto, de forma que tras la impronta de CP/M, llame Mallard BASIC en la forma

A > BASIC

Introduzca el siguiente listado y sálvelo con SAVE "INICIAR"

```
10 cls$=CHR$(27)+"E"+CHR$(27)+"H"
20 DEF FN locate$(fil,col)=CHR$(27)+"Y"+CHR$(32+fil)+CHR$(32+col)
30 ivideo$=CHR$(27)+"p"
40 nvideo$=CHR$(27)+"q"
50 DELETE
```

Se trata como verá, de añadir unas pequeñas ayudas a la programación. La línea 10 define la variable CLS$ que mediante PRINT producirá un borrado de pantalla y cursor al origen, similar al efecto producido por los de otras versiones BASIC.

La línea 20 define una función que emula el LOCATE de otras versiones y permite mediante FN LOCATE$ (F, C) imprimir en texto en la fila y columna indicados.

IVIDEO$ y NVIDEO$ son dos variables que permiten invertir el color de la pantalla y devolverlo a la normalidad respectivamente.

La idea queda abierta, si conoce trucos o herramientas de trabajo que desea utilizar, puede aprovechar y ampliar las posibilidades de este programa (ver apéndice 6).

3. CONSECUENCIAS DE LA INSTALACION

Apague el ordenador, introduzca el disco que ha ido construyendo y que debe contener

```
A>dir
A  BASIC    COM   SETKEYS  COM . LANGUAGE COM   ED        COM : TECLAS   BAK
A  TECLADO  KEY   TECLAS   KEY . INICIAR  BAS · PROFILE  SUB : J14SCPM3 EMS
A  SUBMIT   COM
```

enciéndalo y en pantalla irá obteniendo

A > SET KEYS TECLAS.KEY
A > BASIC INICIAR

A continuación aparecerá el conocido mensaje de Mallard BASIC. Ahora se encuentra en situación de trabajar con él, aprovechándose tanto las variables y funciones definidas en INICIAR.BASIC como las órdenes BASIC que obtendrá del acuerdo con la siguiente tabla

Teclas	*Orden*
ALT + F	FOR
ALT + G	GOTO
ALT + L	PRINT FN LOCATE$ (
ALT + I	INPUT
ALT + T	THEN
ALT + E	ELSE
ALT + C	PRINT CLS$
ALT + N	NEXT
f1	LIST
f2	SAVE "
f3	LOAD "
f4	CHR$ (

Las ventajas son evidentes, pero los inconvenientes no tanto, el mayor es sin duda la pérdida de capacidad del disco. El método es ideal para programar, pero no será tanto si el programa desarrollado utiliza ficheros grandes, que necesita la máxima capacidad. En estos casos utilice este disco exclusivamente para cargar el BASIC y salve sus programas y ficheros en otros discos.

CAPITULO 2

MATRICES

1. CONCEPTOS PREVIOS

2. MATRICES NUMERICAS Y ALFANUMERICAS

3. DIMENSIONES DE UNA MATRIZ
* Ejercicios básicos sobre matrices

4. CARGAR UNA MATRIZ
* Asignación con INPUT
* Asignación con READ-DATA
* Asignación desde ficheros

5. ERRORES EN EL MANEJO DE MATRICES

6. OPERACIONES MAS FRECUENTES CON MATRICES
* Suma de los elementos de una lista
* Suma de los elementos de una tabla
* Búsqueda del mayor o menor de una lista
* Corregir datos de una matriz
* Anular un elemento en una lista
* Añadir elementos a una lista
* Búsqueda de elementos alfanuméricos de listas y tablas

7. EJERCICIOS RESUELTOS
1) Notas medias de un curso
2) Producto de una matriz por un valor
3) Inversa de una matriz
4) Determinante de una matriz cuadrada
5) Sistemas de ecuaciones lineales

2. Matrices

1. CONCEPTOS PREVIOS

1.1. Las variables

Como conocerá, el PCW al igual que otros ordenadores, permite trabajar con varios tipos de variables, formando dos grandes grupos: simples e indexadas (matrices).

Las variables simples, son sin duda las más intuitivas y su característica principal es la de poder almacenar en ella un solo dato, sin relación con otros. Por el contrario las indexadas permiten almacenar datos relacionados entre sí. Dado que las variables simples son de fácil manejo, se dedica este capítulo a las indexadas, por cuanto su conocimiento, suele estar mucho menos extendido.

En el gráfico siguiente se han sintetizado todos los tipos de variables existentes, símbolo que las caracteriza y rango de existencia, con el fin de permitir, de forma clara, una visión de conjunto sobre las mismas.

- VARIABLES
  - SIMPLES
    - NUMERICAS
      - ENTERAS (%) — Rango (—32 768 a 232 768)
      - REALES
        - SENCILLAS (!) — 7 cifras significativas (-1×10^{38} a 1×10^{38})
        - DOBLE PRECISION (#) — 16 cifras significativas
    - ALFANUMERICAS ($) — Rango (0 a 255 caracteres)
  - INDEXADAS ( )
    - NUMERICAS
      - ENTERAS
      - REALES
        - SENCILLAS
        - DOBLE PRECISION
    - ALFANUMERICAS

Como puede observarse, a efectos de clasificación, los dos grandes grupos se subdividen de igual forma, por tanto existen igual número de tipos de variables simples, que de indexados.

A continuación se describen someramente todos los tipos.

* Variables numéricas. Son aquellas que permiten almacenar números y pueden ser de dos tipos.

— Enteras. Almacenan sólo números enteros (números sin parte decimal, si se asignan números con decimales, estos son redondeados al entero mas próximo). Se identifican con el símbolo %.

— Reales. Almacenan sólo números reales (números con o sin decimales). Según su precisión se subdividen en dos.

- Sencillas. Los números son tratados en precisión sencilla, que arroja siete cifras significativas, es decir, siete cifras exactas. Se identifican con !.
- Doble precisión. Los números se tratan con precisión mayor, de forma que se disponen de dieciseis cifras significativas. Se identifican con #.

* Variables alfanuméricas. Almacenan cadenas de caracteres (números, letras y símbolos). A pesar de que una cadena puede estar formada sólo por números, su caracter es el de "texto" y no numérico propiamente dicho.

Esta descripción es válida tanto para las variables simples, como para la indexadas.

1.2. Variables indexadas o matrices

Como ya se ha mencionado las variables simples permiten tratar datos uno a uno, de forma independiente, pero existen muchas ocasiones, en que es necesario tratar conjuntos de datos que aún siendo distintos entre si, tienen un nexo común, por ejemplo del conjunto formado por todas las estaturas de los alumnos de una clase, el tratamiento de estos datos de naturaleza homogénea debe permitir una manipulación como conjunto además de como elementos aislados, de aquí surge la necesidad de disponer de variables unidas por un lazo común y estas reciben el nombre de variables indexadas o matrices.

El ejemplo que sigue ilustra la necesidad mencionada

CLASE Nº A

Alumno	*Estatura*
1	1.50
2	1.54
3	1.52
4	1.61
5	1.49
.	.
.	.
.	.
20	1.62

Utilizando variables simples, haríamos corresponder a la var A = 1,50 a la B = 1,54 y así sucesivamente. Si tuviésemos que calcular la estatura media, la expresión sería

$$M = (A + B + C + \ldots) /20$$

Este tipo de cálculo resulta muy largo y pesado y por tanto inapropiado, sin embargo si consideramos la posibilidad de que existan unas variables de igual nombre, identificadas con un índice (1 a 20) que hagamos corresponder con el número de orden del alumno el proceso de cálculo podrá regirse por un bucle FOR que varíe de 1 a 20. Esta es la idea que se persigue y que ha dado origen a las variables indexadas, conocidas también como matrices por la analogía con dicha estructura matemática.

(1.50, 1.54, 1.52, 1.61, 1.49, ..., 1.62) Matriz de estaturas
(A1, A2, A3, A4, A5, ..., A20) Matriz de variables

El estudio de las matrices, resulta pues fundamental en programación, ya que permite un cómodo, rápido y eficaz tratamiento de bloques de información.

1.3. Concepto informático de matriz

Una matriz es un bloque o conjunto de celdas reservadas en la memoria del ordenador, que pueden ser referenciadas con un nombre simple y común a todos los elementos del conjunto, y distinguidas mediante un número (índice) de orden, que señala a una celda en particular.

Gráficamente estas estructuras nos recuerdan a las matrices matemáticas y de ahí su nombre, aunque su contenido y manejo se aparta del concepto matemático. Por ejemplo, un bloque formado por 5 celdas puede representarse de dos formas:

Horizontal

1	2	3	4	5

Matriz fila (vector)

Vertical

1
2
3
4
5

Matriz columna (vector)

Si llamamos A, a cualquiera de las matrices anteriores, el nombre de sus elementos sería

$$a_1, a_2, a_3, a_4 \text{ y } a_5$$

es decir, el nombre es común y se identifican con un subíndice que representa el orden (1, 2, 3, 4 y 5).

En BASIC los índices se escriben entre paréntesis, de forma que el ejemplo se expresaría

a(1), a(2), a(3), a(4) y a(5)

Como veremos a continuación, también pueden reservarse bloques de dos dimensiones e incluso de más; por ejemplo:

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

Bloque de 3 × 5

Los índices de una matriz de dos dimensiones, no corresponden a la numeración expuesta en el ejemplo anterior, sino que se utiliza un índice doble (2 números), el primer número indica la fila a que pertenece y el segundo, la columna. En el ejemplo anterior sería:

COLUMNAS

FILAS	1ª	2ª	3ª	4ª	5ª
1ª	1.1	1.2	1.3	1.4	1.5
2ª	2.1	2.2	2.3	2.4	2.5
3ª	3.1	3.2	3.3	3.4	3.5

El elemento de índice 2.4, sería el situado en la fila 2ª y columna 4ª.

Por último, nos falta el nombre. Como indicamos anteriormente, el nombre es común para todo el conjunto, por tanto, si llamamos A al conjunto del ejemplo último, sus elementos serán conocidos como A11, A12, A13... A35, que es la nomenclatura matemática. En BASIC, los nombres de los elementos son A(1,1); A(1,2); A(1,3); ....; A(3,5); es decir, la sintaxis general es

Nombre de la matriz (común a todos sus elementos)

A (F , C)

columna

fila

Consideraciones

— El nombre de una matriz puede ser cualquiera que sea válido para variables en el BASIC del Amstrad, por ejemplo:

NOMBRES DE MATRICES

Válidos	*No válidos*
A	78 Z
HOLA	07
B7	4B

— Los subíndices comienzan su numeración en cero (0) o en uno (1). (Ver epígrafe dedicado a la sentencia DIM).

— Los subíndices, en su sentido más amplio, pueden ser:
  — Números (p.e. 4)
  — Variable numérica (p.e. VEL)
  — Expresión numérica (p.e. A*4/2)

  Tanto las variables numéricas como los resultados de las expresiones, contendrán o darán como resultado números enteros positivos (naturales).

— Lista.—Se denomina lista a una matriz de una sola dimensión (vector).

— Tabla.—Se conoce como tabla a las matrices de dos dimensiones.

— El número de elementos de una matriz viene determinado de forma explícita por la orden DIM y de forma implícita se toma 10.

2. MATRICES NUMERICAS Y ALFANUMERICAS

Al igual que en las variables simples, existen matrices numéricas y alfanuméricas, dependiendo de su contenido:

Tipo	*Contenido*	*Nombre*
Numéricas	Números	válido en BASIC (*)
Alfanuméricas	Cadenas	válido en BASIC seguido del símbolo $

(*) Se utilizarán los símbolos #, !, %, si se omite se toma ! (ver primer epígrafe del capítulo)

Respecto al nombre, son válidas todas las reglas de uso en las variables simples. Debe hacerse notar que para el ordenador, las variables siguientes son distintas:

A 1 - Variable simple numérica
A 1 $ - Variable simple alfanumérica
A (1) - Elemento de la matriz numérica A
A $ (1) - Elemento de la matriz alfanumérica A $

Estas cuatro variables pueden coexistir perfectamente en un programa, por cuanto para el ordenador son totalmente distintas. De todos modos, no resulta una práctica

aconsejable, pues la confusión viene por parte del programador y siendo tan amplia la posibilidad de la elección de nombres, el utilizar esta técnica resulta poco razonable, a menos que deliberadamente se desee programar con algo de ocultismo.

Como ejemplo de ambos tipos, representamos gráficamente el contenido de dos listas, una numérica y otra alfanumérica.

Nombre de la matriz	*Nº de elementos*	*Nombre y contenido de los elementos*			
DAT	4	DAT(0)	DAT(1)	DAT(2)	DAT(3)
		6	2.52	45	0

VINO$	3	VINO$(0)	VINO$(1)	VINO$(2)
		RIOJA	VALDEPEÑAS	JEREZ

Notas:

- Se considera en el ejemplo el cero (0) como primer índice.
- Las matrices numéricas sólo pueden contener números.
- Las matrices alfanuméricas, además de textos o palabras propiamente dicho, pueden contener números, pero éstos serán considerados como palabras (conjunto de caracteres) y no como tales números.
- Las denominaciones texto, palabra o cadena se utilizan habitualmente con el mismo significado: conjunto de caracteres (en inglés string).
- Los elementos de las matrices numéricas están sujetos a las reglas y funciones matemáticas propias de los números y variables numéricas.
- Los elementos de las matrices alfanuméricas únicamente podrán utilizarse con las reglas y funciones propias del BASIC para el manejo de palabras (strings).

A continuación exponemos un cuadro, no exhaustivo, de operaciones y funciones válidas y no válidas con elementos de matrices, a título de ejemplo:

Válido	*No válido*
A(1) * A(8)/2	A$(4)/2
A$(3) + Z$(0)	VAL (A(7))
X$(1) + "PEPE"	A$(0) + B(4) + 38
VAL (X$(1))	LEN (A(4))
LEFT$ (Z$,4)	MID$ (A(4), 6, 8)
A(I) * Z(K)	(I Y K VAR NUM

3. DIMENSIONES DE UNA MATRIZ

Hasta ahora hemos visto conceptos generales sobre las matrices, bajo el punto de vista informático y en concreto dentro del lenguaje BASIC.

Veamos ahora cómo se utilizan desde el BASIC del Amstrad (aunque únicamente nos referimos a los Amstrad PCW 8256 y PCW 8512, suele ser común a la mayoría de los ordenadores).

La sentencia DIM

La sentencia DIM se utiliza para establecer el número de elementos de una matriz, de tipo numérico o alfanumérico.

La síntaxis general es

DIM nombre (N)

nombre: nombre válido para variables
N: número de elementos (dimensión)

Así. DIM A(4) reserva 5 espacios de memoria para poder guardar en ellos los valores de los elementos de la matriz A, que serán A(0), A(1), A(2), A(3) y A(4).

En el caso de matrices con dos dimensiones, la síntaxis es:

DIM nombre (F,C)

siendo F el número de filas —1 y C el número de columnas —1.

Por ejemplo: DIM A(2,3) reservará (2 + 1) * (3 + 1) = 12 espacios de memoria, cuyos elementos se pueden representar gráficamente como:

A(0,0)	A(0,1)	A(0,2)	A(0,3
A(1,0)	A(1,1)	A(1,2)	A(1,3)
A(2,0)	A(2,1)	A(2,2)	A(2,3)

La presencia del elemento cero, resulta a veces, para los programadores principiantes, perturbador, sobre todo en las matrices bidimensionales. Resulta práctica común hacer abstracción de todos los elementos cero y tratar la anterior matriz como si estuviese sólo compuesta por:

A(1,1)	A(1,2)	A(1,3)
A(2,1)	A(2,2)	A(2,3)

Aunque lícito, evidentemente se desaprovecha memoria. Para resolver este problema, se dispone de la sentencia OPTION BASE, que va seguida de un parámetro, el cual toma únicamente los valores cero o uno (0 y 1).

— OPTION BASE 0. Hace que todas las matrices que se dimensionen tengan como primer índice el cero (0).

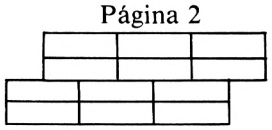
— OPTION BASE 1. Los índices comenzarán en uno (1).
Esta sentencia debe utilizarse antes del dimensionamiento, pues en caso contrario produce error. Además sólo puede utilizarse una sola vez en el mismo programa.

Así pues puede elegirse el comienzo de los índices en la forma que mejor se adapte a las necesidades concretas de cada programa. Como podrá verse más adelante a veces resulta útil el elemento cero para almacenar en él datos relativos al resto de los elementos, por ejemplo la suma, la media, etc.

Hasta el momento han aparecido matrices de una y dos dimensiones, pero en términos generales, pueden existir matrices de N dimensiones. Por ejemplo: una matriz tridimensional podía ser:

```
DIM A(2,3,2)
```

y estaría formada por 2 filas, 3 columnas y 2 páginas (sin contar los elementos 0, a fin de simplificar), esta matriz podía representarse gráficamente en la forma siguiente.



Página 1

A(1,1,1)	A(1,2,1)	A(1,3,1)
A(2,1,1)	A(2,2,1)	A(2,3,1)

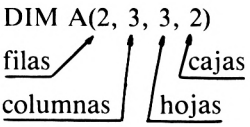
A(1,1,2)	A(1,2,2)	A(1,3,2)
A(2,1,2)	A(2,2,2)	A(2,3,2)

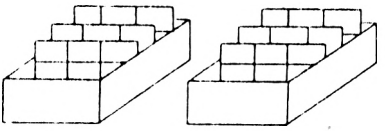
Página 2

En realidad y a la vista del gráfico, puede considerarse que una matriz de tres dimensiones, está compuesta por dos matrices bidimensionales, donde el último índice representa la matriz a que pertenece el elemento.

Las matrices de cuatro dimensiones son algo más complicado de imaginar, y el grado de dificultad hace que sean poco operativas. Para mayores dimensiones 5, 6, etc., la estructura y manejo resultan muy difíciles y por tanto poco utilizadas.

Una representación gráfica de las cuatro dimensiones, pudiera ser la de cajas, conteniendo hojas formadas por matrices bidimensionales.





Algún lector imaginativo puede hacer ejercicios mentales, considerando que las de cinco dimensiones son archivadores, llenos de cajas, que contienen fichas de tablas (matrices de 2 dimensiones). Y las de 6 habitaciones que contienen archivadores, que...

¿Hasta qué nivel se considera capaz de concebir? ¿Y de manejar? Esta última respuesta puede contestarla después.

Consideraciones sobre DIM

— En una misma sentencia DIM pueden agruparse varias matrices para su dimensionamiento.

La síntaxis general es:

DIM nombre 1 (dimensión), nombre 2 (dimensión), ...

Por ejemplo:

DIM A(4), A$(3,2), VIN (100,100)

dimensiona simultáneamente las 3 matrices A, A$ y VIN.

— Si el índice de una matriz no va a superar el valor 10, no es necesario su previo dimensionamiento. (declaración implícita).

— La dimensión de una matriz está dada generalmente por números, pero también es válido dimensionar según el valor de una variable, o incluso, una expresión numérica

Dimensión de una matriz { Número / Variable numérica / Expresión numérica

Por ejemplo:

1) 10 INPUT "Número de alumnos"; N
   20 DIM AL$(N)

2) 10 INPUT "NUMERO DE FICHAS";NUM
   20 DIM F$(NUM*2,4)

Si la variable o la expresión no son números enteros, se toma como dimensión el entero más próximo, así:

10 N = 3.2
20 DIM A(N*2)

La dimensión de A será la parte entera de 3.2*2 = 6.

— Si por alguna razón, una matriz no va a volver a utilizarse como tal y los valores contenidos no son de ningún interés, puede liberarse el espacio reservado en memoria mediante la orden ERASE.

La sintaxis general es:

ERASE nombre 1, nombre 2, ...

Donde nombre representa a las matrices a eliminar; por ejemplo:

```
10 DIM A(10), B$(5), Z(100)        Crea 3 matrices A, B$ y Z
.
.
.
210 ERASE A, Z                     Elimina las matrices A y
                                   Z, liberando el espacio
                                   ocupado por ellas.
```

Una vez eliminada una matriz, puede volver a utilizarse su nombre para crear una nueva; tarea que se denomina redimensionamiento y que sólo puede realizarse después de un ERASE.

Siguiendo con el ejemplo, sería válida una línea como la siguiente:

```
.
.
.
480 DIM A(24)
```

Ya que la matriz A fue borrada con anterioridad.

Si se utiliza ERASE y la matriz a borrar no fue declarada implícita o explícitamente (DIM), se produce un mensaje de error.

4. CARGAR UNA MATRIZ

Genéricamente, se denomina cargar una matriz a la operación por la cual se asigna a cada uno de sus elementos un valor.

Cuando se utiliza la orden DIM, automáticamente el ordenador asigna un cero a todos los elementos o una cadena vacía ('''') en las alfanuméricas.

Por ejemplo:

DIM A(3)

A(0)	A(1)	A(2)	A(3)
0	0	0	0

Elementos

DIM B$(2)

A$(0)	A$(1)	A$(2)
''''	''''	''''

Las dobles comillas representan a una *cadena vacía*, es decir, una palabra formada por "ningún carácter", ni siquiera el blanco.

La tarea de carga o asignación viene regida generalmente por bucles FOR, encargados de la manipulación de los elementos de las matrices, aprovechándose la variable del FOR como variable índice de la matriz. Para más de una dimensión se utilizan los bucles anillados, tantos como dimensiones tenga la matriz.

Asignación con LET

Dado que LET es en el BASIC la sentencia de asignación por antonomasia, nos referimos a ella en primer lugar.

La forma elemental será;

LET variable indexada = valor (numérico o alfanumérico)

Ejemplos:

1) Valores distintos

```
10 DIM A(2)
20 LET A(0) = 4
30 LET A(1) = 6
40 LET A(2) = 9
```

A(0)	A(1)	A(2)
0	0	0
4	6	9

2) Con los mismos valores para todos los elementos (inicializar)

```
10 DIM A$(3)
20 FOR I=0 TO 3
30 LET A$(I) = "PEPE"
40 NEST I
```

A$(0)	A$(1)	A$(2)	A$(3)
PEPE	PEPE	PEPE	PEPE

El uso de LET es muy variado y su potencia, estriba en utilizar otras variables y no números en la asignación. También es útil cuando se desea inicializar una matriz con todos sus valores iguales, incluso poniéndolos a cero, pues debe tenerse en cuenta que la sentencia DIM sólo puede utilizarse una sola vez para la misma matriz; por tanto, si queremos restaurar en un momento dado las condiciones iniciales, necesitaremos un bucle como el del ejemplo 2, para la puesta a cero (o vacío) de la matriz, aunque también podemos borrar la matriz con ERASE y volver a dimensionar con DIM.

Asignación con INPUT

En determinadas ocasiones es necesario introducir los datos por teclado, para lo cual utilizamos la sentencia de entrada INPUT.

También es posible utilizar INKEY$ e incluso el joystick, pero su aplicación es más restringida.

Veamos con un ejemplo, el uso de INPUT:

```
10 DIM A(4)
20 FOR I=0 TO 4
30 INPUT A(I)
40 NEXT I
```

En una asignación como la indicada, cuando el número de datos a introducir es elevado, es frecuente que se produzcan errores como consecuencia de la pérdida del orden en dicha entrada. Para evitarlos, es suficiente el introducir previamente al INPUT un PRINT que nos indique el número de orden del dato en curso. Como ejemplo, añada al aprograma anterior la línea:

```
25 PRINT "DATO";I;
```

Haga nuevamente RUN y observe el efecto. El punto y coma al final de la línea 25 permite que la interrogante del INPUT aparezca a continuación del PRINT, sin saltar línea.

En el caso de tener que introducir cadenas alfanuméricas (textos), que contengan comas, debe utilizarse LINE INPUT, que permite introducir una línea completa de caracteres, cualesquiera que sea su contenido.

Si las cadenas a introducir tienen un número fijo de caracteres, puede utilizarse INPUT$.

Veamos un ejemplo. Si deseamos tener en una matriz los días de la semana, de forma que al 1 le corresponda el lunes, al 2 el martes, etc... podemos utilizar el siguiente procedimiento:

```
10 DIM DS$(7)
20 FOR I = 1 TO 7
30 READ DS$(I)
40 NEXT I
50 DATA LUNES, MARTES, MIERCOLES, JUEVES, VIERNES,
SABADO, DOMINGO
```

Se observa que el bucle se establece desde 1 y no desde 0, a fin de hacer corresponder al lunes el índice 1, premisa del ejercicio. El elemento D$(0) existirá y contendrá el vacío (ningún carácter), ya que no se ha utilizado OPTION BASE y por omisión, como recordará, el ordenador toma el valor cero como primer índice.

Dado que el orden de la matriz es menor que 10, no es estrictamente necesario su dimensionamiento.

Si desea visualizar los datos puede añadir

```
35 PRINT DS$(I)
```

Los bucles de lectura, como el expuesto en éste y anteriores ejemplos, resulta conveniente agruparlos, de forma que desde el FOR al NEXT todo se encuentre en la misma línea, ya que son operaciones relacionadas entre sí.

Por ejemplo:

```
20 FOR I = 1 TO 7 : READ DS$(I) : NEXT
```

Deliberadamente no incluimos la variable I del FOR, en la sentencia NEXT, pues en su Amstrad no es necesario, aunque a veces resulta más conveniente su uso por razones de claridad.

Asignación desde ficheros

Dado que este tema será tratado con amplitud, en los siguientes capítulos, únicamente se indica, que las matrices también pueden cargarse con datos contenidos en un fichero y de hecho, esta práctica, es tanto o más común que las indicadas anteriormente.

Al final de su manual encontrará una lista de los errores emitidos por BASIC, de ellos solo dos corresponden exclusivamente al manejo de matrices. El apéndice 3 de este libro está dedicado a este tema.

5. ERRORES EN EL MANEJO DE MATRICES

1) Subscript out of range (error 9)

   El subíndice utilizado es demasiado grande o demasiado pequeño. El límite inferior es 0 y el superior viene dado por la sentencia DIM correspondiente y tiene como límite la capacidad de la memoria RAM disponible.

   Ejemplo:

```
10 DIM A (8)
20 A(5) = 24
30 A(9) = 6
40 H = —2
50 A(H) = 4
```

   Las líneas 30 y 50 dan error, puesto que —2 y 9 están fuera del rango de la matriz

2) Array already dimensioned (error 10)

   La matriz que trata de dimensionarse ya lo estaba anteriormente. No se puede redimensionar una matriz, ni aún con la misma dimensión, salvo que previamente haya sido borrada con ERASE.

   Ejemplo:

```
10 DIM A(10)
.
.
.
180 DIM A(10) o 180 DIM A(24)
```

   Error por redimensionamiento de la matriz A

Otros tipos de error pueden surgir, pero afectarán a las variables, sintaxis, límite de memoria, etc., pero no a las matrices propiamente.

6. OPERACIONES MAS FRECUENTES CON MATRICES

Suma de los elementos de una lista

En muchas ocasiones es necesario obtener la suma de todos los elementos de una lista numérica (matriz de 1 dimensión).

Para ello debe utilizarse —como siempre— un bucle, cuya variable sirva como índice de los elementos de la matriz y de una variable simple que llamaremos SUMA, en la cual almacenaremos el valor de la suma de todos los elementos.

El programa tendrá tres partes: la primera de *carga*, por cualquiera de los métodos expuestos, la segunda de *suma*, y la tercera, de *salida* por pantalla del resultado.

```
10 ' CARGA DE DATOS
20 DIM A(6)
30 FOR K=0 TO 6
40 READ A(K)
50 NEXT K
60 ' SUMA DE DATOS
70 FOR K=0 TO 6
80 SUMA=SUMA+A(K)
90 NEXT K
100 ' SALIDA DE RESULTADOS
110 PRINT "LA SUMA DE LOS NUMEROS"
120 FOR K=0 TO 6
130 PRINT A(K);"+";
140 NEXT K
150 PRINT "=";SUMA
200 DATA 4,3,6,12,45,8,2
```

Es interesante volver a insistir en la posibilidad de almacenar la suma en el elemento cero de la matriz, eliminando la variable simple SUMA.

Observe que como son 7 los datos, el nuevo dimensionamiento será precisamente DIM A(7), ya que el elemento A(0) queda reservado para la suma. Los límites de los bucles no serán 0 y 6, sino 1 y 7, respectivamente.

Veamos cómo quedaría el programa tras introducir las modificaciones apropiadas.

```
10 ' CARGA DE DATOS
20 DIM A(7)
30 FOR K=1 TO 7
40 READ A(K)
```

```
50 NEXT K
60 ´ SUMA DE DATOS
70 FOR K=1 TO 7
80 A(0)=A(0)+A(K)
90 NEXT K
100 ´ SALIDA DE RESULTADOS
110 PRINT "LA SUMA DE LOS NUMEROS"
120 FOR K=1 TO 7
130 PRINT A(K);"+";
140 NEXT K
150 PRINT "=";A(0)
200 DATA 4,3,6,12,45,8,2
```

Aunque la carga se ha realizado mediante READ y DATA puede utilizarse cualquier sistema de los indicados anteriormente.

Suma de los elementos de una tabla

Con las tablas (matrices de dos dimensiones) se presentan tres casos distintos:

* Suma de elementos por filas
* Suma de elementos por columnas
* Suma de todos los elementos

Veamos con un ejemplo los tres casos. Sea la tabla

						Σ FILAS
	2	6	5	1	14	
	4	—3	2	0	3	
	3	5	1	—1	8	
Σ COLUMNAS	9	8	8	0	25	Σ TOTAL

El programa cargador, común a los tres cálculos, podía ser (supuestos los datos de la tabla contenidos en líneas DATA):

```
10 ´ CARGAR LA MATRIZ
20 OPTION BASE 1
30 DIM A(3,4)
40 FOR F=1 TO 3
50 FOR C=1 TO 4
60 READ A(F,C)
```

```
70 PRINT A(F,C),
80 NEXT C
90 PRINT
100 NEXT F
110 DATA 2,6,5,1,4,-3
120 DATA 2,0,3,5,1,-1
```

*** Suma de todos los elementos**

La suma de todos los elementos no reviste ningún problema, se establecen dos bucles iguales a los de lectura y en la variable simple TOTAL se van acumulando uno a uno el valor de todos los elementos.

```
130 ' SUMA DE TODOS LOS ELEMENTOS
140 FOR F=1 TO 3
150 FOR C=1 TO 4
160 TOTAL=TOTAL+A(F,C)
170 NEXT C,F
```

Observe que la expresión: TOTAL = TOTAL + A(F,C) no debe interpretarse en sentido algebraico, puesto que ello equivaldría a decir que A(F,C) = 0, ya que si despejamos en la expresión, tenemos TOTAL — TOTAL = A(F,C). Por tanto, el sentido es que el nuevo valor de la variable TOTAL ha de ser igual al valor antiguo más el valor del enemigo A(F,C).

*** Suma de filas**

Dado que el número de filas es tres, necesitaríamos tres variables simples para almacenar la suma de los elementos de cada una de ellas y como consecuencia, tres bucles, uno por fila.

Pero observando las líneas 200 a 240, estructura de dos bucles, y considerando que el primero controla las filas, se deduce que sería más útil considerar una matriz unidimensional de tres elementos, cuyo índice viene controlado precisamente por el bucle de las filas.

Si llamamos SF (suma de filas) a la matriz, el programa sería:

```
180 ' SUMA POR FILAS
190 FOR F=1 TO 3
200 FOR C=1 TO 4
210 SF(F)=SF(F)+A(F,C)
220 NEXT C,F
```

La matriz SF no se dimensiona previamente, puesto que sus elementos no superan el límite de 10.

*** Suma de columnas**

La tercera parte del cálculo, suma de columnas, es igual a la anterior. También utilizamos una matriz sin previo dimensionamiento para almacenar las sumas de las cuatro columnas.

La diferencia con la anterior estriba en que los bucles están cambiados; primero el de columnas y luego el de filas.

```
230 'SUMA POR COLUMNAS
240 FOR C=1 TO 4
250 FOR F=1 TO 3
260 SC(C)=SC(C)+A(F,C)
270 NEXT F,C
```

Por último, procedemos a sacar por pantalla todos los datos en forma de tabla, tal como se planteó al principio.

```
280 ' SALIDA DE RESULTADOS EN FORMA DE TABLA
290 FOR  F=1 TO 3
300 FOR C=1 TO 4
310 PRINT A(F,C);
320 NEXT C
330 PRINT "*";SF(F)
340 NEXT F
350 PRINT "****************"
360 FOR C=1 TO 4
370 PRINT SC(C);
380 NEXT C
390 PRINT "*";TOTAL
```

El programa ya está completo, haga RUN para comprobar y obtendrá una tabla con las sumas de las filas, de las columnas y la suma total, tal como se indica a continuación.

2	6	5	1	*	14
4	—3	2	0	*	3
3	5	1	—1	*	8

****

9	8	8	0	*	25

Búsqueda del mayor o menor de una lista

* Números

Antes de entrar en el campo de la ordenación, como introducción, veamos cómo encontrar el mayor o el menor elemento de una lista o tabla.

Utilizando los datos del ejemplo M1 y las líneas 10 a 50 y la 200, que componen la carga de una lista en la matriz A, añadimos las siguientes líneas para la búsqueda del mayor.

```
10 ' CARGA DE DATOS
20 DIM A(6)
30 FOR K=0 TO 6
40 READ A(K)
50 NEXT K
200 DATA 4,3,6,12,45,8,2
300 ' BUSQUEDA DEL MAYOR
310 FOR K=0 TO 6
320 IF A(K)>MAYOR THEN MAYOR=A(K)
330 NEXT K
340 PRINT "EL MAYOR DE LOS NUMEROS"
350 FOR K=0 TO 6
360 PRINT A(K);
370 NEXT K
380 PRINT "ES EL";MAYOR
```

La línea 310 es la encargada de ver quién es el mayor en cada vuelta del bucle, que maneja los siete elementos de la lista.

La variable MAYOR contiene el número mayor encontrado hasta una determinada vuelta del bucle.

Cuando K = 0 el valor de MAYOR = 0, por tanto en la primera vuelta MAYOR = A(0), si el siguiente valor es mayor se producirá un cambio, en caso contrario, continúa el mismo valor como mayor.

Veamos los valores que en el ejemplo toman las variables que intervienen.

Vuelta	*K*	*A(K)*	*MAYOR*
1	0	4	4
2	1	3	4
3	2	6	6
4	3	12	12
5	4	45	45
6	5	8	45
7	6	2	45

La búsqueda del menor es análoga, basta con cambiar el símbolo > de la línea 320 por el de <, y el nombre de la variable MAYOR por otro más apropiado.

Si los números con los que se va a trabajar son menores que cero (alguno), bastará con iniciar la var MENOR de forma implícita, pero si todos son positivos, el valor inicial explícito de MENOR será mayor que el más pequeño, como práctica puede darse a MENOR = 9999999 o número análogo.

Por ejemplo

```
  10 ' CARGA DE DATOS
20 DIM A(6)
30 FOR K=0 TO 6:READ A(K):NEXT K
40 DATA 4,6,2,7,23,12,5
50 ' BUSQUEDA DEL MENOR
60 MENOR=9999
70 FOR K=0 TO 6
80 IF A(K)<MENOR THEN MENOR=A(K)
90 NEXT K
100 PRINT "EL MENOR ES";MENOR
```

*** Cadenas (palabras)**

El concepto de mayor y menor, aplicado a caracteres, es distinto del numérico.

Los caracteres están ordenados de menor a mayor en la forma alfabética, es decir A, B, C...., Z. dicha ordenación no es arbitraria, sino que corresponde a un código estándar, que en el caso del Amstrad (y de la mayoría de los ordenadores), es el código ASCII.

En el apéndice se incluye una tabla completa de dicho código. Como se puede comprobar, a las letras mayúsculas del alfabeto les corresponde un valor que en números decimales comienza con el 65 para la A y finaliza con el 90 para la Z. (Ver Tomo I, pág. 116 del manual).

ASCII	Carácter	A	B	C	D	.....	Y	Z
	Valor decimal	65	66	67	68	.....	89	90

Ahora bien, observará que las letras minúsculas tienen valores distintos, y es que en realidad, son distintos caracteres para el ordenador. Veamos dichos valores:

ASCII	Carácter	a	b	c	d	.....	y	z
	Valor decimal	97	98	99	100	.....	121	122

Una letra (o carácter) es mayor que otra, si su valor ASCII lo es.

C > A ya que 67 > 65

Ahora bien, ¿qué ocurre si comparamos A mayúscula y a minúscula?

Dado que a vale 97 y A vale 65, resulta que:

a > A

La conclusión inmediata es que si establecemos comparaciones, debemos hacerlo con letras de un mismo tipo, mayúsculas o minúsculas, pero no con ambas.

Practique con el siguiente programa y vea los resultados.

```
10 INPUT "Introduzca una letra ";L1$
20 INPUT "Introduzca otra letra";L2$
30 IF L1$>L2$ THEN PRINT L1$;" es mayor que ";L2$:GOTO 50
40 PRINT L2$;" es mayor que ";L1$
50 PRINT "-----------------------"
60 GOTO 10
```

¿Qué ocurre con las cadenas? Veámoslo de forma práctica. Utilice el programa anterior modificando (si lo desea) el texto de los INPUT, sustituyendo LETRA por PALABRA y observe los resultados obtenidos introduciendo palabras en mayúsculas, en minúsculas o mixtas.

En la tabla siguiente incluimos algunos ejemplos que nos permitirán sacar consecuencias para nuestros propósitos.

PALABRA (A)	*PALABRA (B)*	*RESULTADO*
1 TOLEDO	Toledo	A < B
2 ZAPATO	abanico	A < B
3 PERA	PERO	A < B
4 124	125	A < B
5 cara	caRa	A > B

1. TOLEDO es menor que Toledo, puesto que las letras minúsculas tienen un valor ASCII mayor.
2. ZAPATO es menor que abanico por igual motivo, a pesar de que zapato comienza con la última letra del abecedario y abanico con la primera.
3. Puesto que las tres primeras letras son iguales, la decisión de ser PERO < PERA, es debido únicamente a la letra O frente a la A.
4. Los números tratados como caracteres también responden al concepto matemático de mayor o menor.
5. cara es mayor que caRa puesto que r es mayor que R, en términos ASCII.

Hagamos ahora un programa que nos permite hallar la cadena menor de una lista o, lo que es lo mismo, la cadena que alfabéticamente es la primera.

```
10   MENOR DE UNA LISTA ALFANUMERICA
20 ´ Primero alfabeticamente
30 DIM A(9): ´ 10 Elementos
40 ´ ........ CARGA ........
50 FOR K=0 TO 9
60 READ A$(K)
70 NEXT K
80 ´ .......BUSQUEDA .......
90 MENOR$="Z"
100 FOR K=0 TO 9
110 IF A$(K)<MENOR$ THEN MENOR$=A$(K)
120 NEXT K
130 CLS:PRINT "LA MENOR DE LAS PALABRAS"
140 FOR K=0 TO 9:PRINT A$(K):NEXT K
150 PRINT "ES:   ";CHR$(24);MENOR$;CHR$(24)
160 ´ ...LINEAS DE DATOS ...
170 DATA PERA,COCHE,RIO,ZAPATO,RELOJ
180 DATA ARBOL,CEPA,MESA,FLOR,LAPIZ
```

La variable auxiliar MENOR$, ha de inicializarse con un valor alto (letras altas) con el fin de asegurar un buen funcionamiento del programa, puesto que en caso contrario, la variable contendría —por omisión— la cadena vacía, que es evidentemente un valor más bajo que cualquier palabra.

Por lo demás, el programa es similar al utilizado anteriormente para los números.

En el caso de utilizar fechas, tomadas como cadenas, estos sistemas no funcionarían, para este caso concreto vea el apartado correspondiente en el capítulo 3. Ordenación.

Corregir datos de una matriz

Cuando el número de datos es elevado, la probabilidad de cometer errores en la entrada por teclado es alta.

Para prevenir estos errores, necesitamos dos herramientas: la primera, una entrada de datos clara, sobre todo en el caso de tablas, de forma que en pantalla se señale

perfectamente la fila y columna en curso. La segunda, un programa depurador que permita corregir los datos erróneos antes de pasar a la fase de procesado de los mismos.

Veamos con un ejemplo cada una de las partes:

Sea una tabla numérica de 5 × 7 datos, donde las filas representan "conceptos de gasto" y las columnas "los días de la semana".

La mejor forma de identificación es, evidentemente, llamar a las cosas por su nombre, de forma que utilizaremos dos matrices auxiliares (alfanuméricas), para almacenar los nombres de los "conceptos" y de los "días" de la semana. Por ser constantes podemos utilizar líneas DATA.

Las matrices serán:

Conceptos	C$ (5)
Días	D$ (7)
Datos	D (5,7)

```
10 ´ RUTINA PARA CORRECCION DE DATOS
20 DIM C$(5),D$(7),D(5,7)
30 ´ CARGAR CONCEPTOS
40 FOR K=1 TO 5:READ C$(K):NEXT
50 ´ CARGAR DIAS
60 FOR K=1 TO 7:READ D$(K):NEXT
70 DATA Alimentacion,Vestido,Transporte,Casa,Varios
80 DATA Lunes,Martes,Miercoles,Jueves,Viernes,Sabado,Domingo
100 ´ ENTRADA DE DATOS
110 ´ Borrar pantalla
120 FOR F-1 TO 5
130 PRINT "Gastos ";C$(F);" correspondientes al "
140 FOR c=1 TO 7
150 PRINT D$(C);
160 INPUT D(F,C)
170 NEXT C
180 PRINT "-----------------------------------------------"
190 NEXT F
```

Nota: A fin de no infrautilizar la memoria debería añadirse la línea 15 OPTION BASE 1

Como puede ver tras hacer RUN, la entrada de datos resulta clara, se indica en pantalla el tipo de gastos y después el día de la semana. De esta forma, conseguimos una introducción ordenada de los datos, no obstante, a pesar de la claridad, el introducir números induce a errores de teclado, así que vamos a añadir la herramienta de depuración necesaria.

```
200 ' DEPURACION DE DATOS
205 ' Borrar pantalla
210 PRINT "DATOS ACTUALES"
220 PRINT "==============="
230 FOR F=1 TO 5
240 FOR C=1 TO 7
250 PRINT D(F,C);
260 NEXT C
270 PRINT
280 NEXT F
290 INPUT "ALGUN DATO ERRONEO (S/N) ";Z$
300 Z$=UPPER$(Z$)
310 IF Z$="N" THEN 500
320 IF Z$="S" THEN 340
330 GOTO 290
340 INPUT "CUANTOS ERRORES ";E
350 FOR K=1 TO E
360 INPUT "INTRODUZCA LA FILA Y LA COLUMNA ";F,C
370 PRINT "VALOR ERRONEO=";D(F,C),
380 INPUT "NUEVO VALOR ";D(F,C)
390 NEXT K
400 GOTO 200
500 ' PROCESADO DE LOS DATOS
```

Las líneas 210 a 280 se encargan de mostrar los datos en forma de tabla, tal y como se introdujeron por el teclado. Si la variable Z$ contiene una "s" por respuesta afirmativa al INPUT, el programa pregunta (340) cuántos errores se han detectado. El bucle que se inicia en 350 pide que se introduzca la fila y la columna del elemento erróneo, muestra después su valor y pide el nuevo.

La línea 400 devuelve el control al principio de la depuración, con el fin de que se muestre la nueva tabla con las correcciones efectuadas, permitiendo volver a corregir o, en caso de estar bien, con contestar N al INPUT de la línea 290, se abandonará la rutina de depuración para continuar el programa (línea 500 en adelante).

Evidentemente, el ejemplo expuesto es simplemente una idea básica susceptible de muchas mejoras. La entrada de datos y la depuración o filtro de los mismos, es una técnica que algunos programadores han convertido en "arte".

La rutina anterior es, como hemos dicho, básica, pero suficiente en la mayoría de los casos, aunque el lector puede y debe modificarla a su manera. Piense que cada programador tiene un "sello" personal de hacer las cosas, incluso un cierto estándar de representación en pantalla.

Anular un elemento de una matriz

Una tabla o una lista está compuesta de elementos generalmente dinámicos, es decir, sujetos a continuas variaciones del contenido de los mismos.

Anular uno de los elementos es una tarea frecuente. Imaginemos la lista como la siguiente:

1	6	7	8	5	9

compuesta de 6 elementos, en la que deseamos eliminar el cuarto elemento de valor 8, es decir, que la nueva lista tendrá 5 elementos en vez de 6.

La rutina de eliminación podía ser algo así como la que exponemos a continuación:

```
10 ´ RUTINA DE ELIMINACION
20 ´ Carga de la lista
30 OPTION BASE 1:N=6: ´No. de elementos
40 DIM A(N)
50 FOR K=1 TO N:READ A(K):NEXT
60 DATA 1,6,7,8,5,9
70 ´ Eliminacion de un elemento
80 INPUT "Que elemento quiere eliminar ";EL
90 FOR K=EL TO N-1
100 A(K)=A(K+1)
110 NEXT K
120 A(N)=0
130 ´Imprimir nueva tabla
140 FOR K=1 TO N:PRINT A(K):NEXT K
```

Una vez conocida la posición del elemento a eliminar, por el INPUT de la línea 80.

Establecemos un bucle cuyos límites son: la posición a eliminar y el número de elementos de la lista menos uno. En realidad, lo que hacemos es un intercambio, asignando al elemento A(K) el valor del siguiente: A(K + 1).

En el ejemplo propuesto, al llegar a la línea 110 la lista en memoria será:

1	6	7	5	9	9

El elemento cuarto que era un 8, ha pasado a ser el siguiente (el 5), y lo mismo ocurre hasta el penúltimo (el N-1). Más ¿qué ocurre con el último? Este no ha variado, así que el 9 ocupa la 5ª posición por el intercambio, y la 6ª por origen. Si no se desea que se repita, sino simplemente que la lista se acorte, hacemos el último elemento igual a cero, como puede verse en el listado, en la línea 120.

La línea 140 nos muestra la nueva tabla.

Como habrá podido adivinar, esta operación de anulación es una de las opciones que debe ofrecer un programa de base de datos, como veremos en los apartados correspondientes.

Queda todavía por ver algo que quizá el lector ya haya echado en falta. Todo lo expuesto está bien, pero ¿qué ocurre si no conocemos la posición del elemento a eliminar, sino su valor? Esto se da mucho en el caso de textos, donde se conoce la palabra a eliminar, pero no su posición y sobre todo, cuando las listas son largas y en las tablas, más difícil de detectar la posición, puesto que ésta viene dada por dos coordenadas: fila y columna.

Este caso presenta dos problemas, la búsqueda del elemento a eliminar y la eliminación. La última parte es idéntica al caso anterior, teniendo en cuenta que la eliminación del último —que antes era poner a cero— ahora será poner en cadena vacía.

Veamos cómo quedaría el ejemplo anterior aplicado a este caso.

```
10 ´ RUTINA DE ELIMINACION (textos)
20 ´ Carga de la lista
30 OPTION BASE 1:N=6: ´No. de elementos
40 DIM A$(N)
50 FOR K=1 TO N:READ A$(K):NEXT
60 DATA platano,pera,manzana,uva,melon,naranja
70 ´ Eliminacion de un elemento
80 INPUT "Nombre del elemento a eliminar ";EL$
90 FOR K=EL TO N
100 IF A$(K)=EL$ THEN EL=K:GOTO 120
110 NEXT K
120 FOR K=EL TO N-1
130 A$(K)=A$(K+1)
140 NEXT K
150 A$(N)=""
160 ´Imprimir nueva tabla
170 FOR K=1 TO N:PRINT A$(K):NEXT K
```

Hay que tener muy en cuenta que el nombre a eliminar debe ser introducido con la misma sintaxis que figura en las DATA, habida cuenta de lo explicado anteriormente sobre mayúsculas y minúsculas.

El bucle 90-110 es el encargado de encontrar el nombre en el momento que se verifica la igualdad de la línea 100, el valor de K indicará precisamente la posición en la lista, por lo que la variable EL recoge dicho valor.

Las líneas 120 a 140 constituyen el bucle de actualización que se inicia en el valor EL (posición de la palabra a borrar) y finaliza en N = 1, al igual que en el programa anterior.

Por último, la línea 150 hace que el último elemento no se repita y desaparezca, acortando la lista en un elemento (el que fue suprimido).

Añadir elementos a una matriz

Este caso es justo el contrario del anterior. En principio, trataremos de introducir un nuevo elemento en una lista, bajo el supuesto de que la matriz es lo suficientemente grande para ello.

La inserción cabe hacerla al final de la lista o en una posición intermedia cualquiera.

El hacer una inserción en cola no supone ningún problema; basta saber el número de elementos cargados que tiene y hacer el siguiente igual al valor a añadir.

Si la posición es intermedia habrá que hacer un hueco y correr —desde ese lugar— un puesto a la derecha al resto de los elementos.

Veamos esto gráficamente con un ejemplo y elaboremos una rutina que haga estas funciones. Sea la lista:

N = 5

2	5	8	7	6	0	0	0

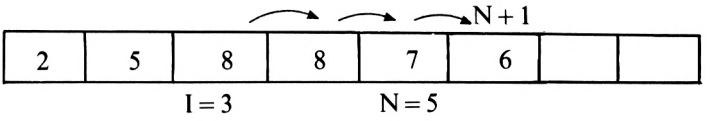
que dispone de 8 elementos, de los cuales los 5 primeros contienen valores y los 3 últimos vacíos (serían 0 por la orden DIM).

Ahora deseamos intercalar un "1" en tercer lugar; es decir, entre el 5 y el 8 de la lista.

```
10 ' INSERCION EN UNA LISTA
20 ' Carga de datos originales
30 OPTION BASE 1:N=5:DIM A(N+3)
40 FOR K=1 TO N:READ A(K):NEXT
50 DATA 2,5,8,7,6
60 ' Insercion
70 INPUT "POSICION DE INSERCION ";I
80 INPUT "VALOR ";V
90 FOR K=N+1 TO I STEP -1
100 A(K)=A(K-1)
110 NEXT K
120 A(I)=V
130 'Salida de la nueva lista
140 FOR K=1 TO N+1:PRINT A(K):NEXT
```

Como puede ver, el bucle de inserción se ha planteado a la inversa que en los ejemplos anteriores. Desde un valor mayor en una unidad, al número de elementos cargados, hasta la posición I de inserción, puesto que N + 1 > I, el valo del paso será —1.

Cuando finalice el bucle, la lista tendrá la siguiente forma:



Por último, sólo falta introducir en la posición I el nuevo valor, que en el ejemplo hemos puesto 1. De ello se encarga la línea 120.

La tabla quedará definitivamente en la forma:

2	5	1	8	7	6		

Como podrá comprobar al hacer RUN en el programa anterior.

La inserción del ejemplo ha supuesto una rotación de elementos a la derecha de la lista. Por supuesto, puede realizarse a la izquierda, basta con modificar los límites del bucle, comenzando en la posición anterior a la primera; es decir, en la cero y finalizando en I—1, y haciendo que A(K) = A(K + 1).

Para listar todos los elementos, el bucle de la 140 tendrá como límite cero en vez de uno y OPTION BASE debe cambiarse para el valor 0 (línea 30).

```
10 ' INSERCION EN UNA LISTA (Por la izquierda)
20 ' Carga de datos originales
30 OPTION BASE 1:N=5:DIM A(N+3)
40 FOR K=1 TO N:READ A(K):NEXT
50 DATA 2,5,8,7,6
60 ' Insercion
70 INPUT "POSICION DE INSERCION ";I
80 INPUT "VALOR ";V
90 FOR K=0 TO I-1
100 A(K)=A(K+1)
110 NEXT K
120 A(I)=V
130 'Salida de la nueva lista
140 FOR K=0 TO N+1:PRINT A(K):NEXT
```

Otro problema que abordaremos más adelante, será la inserción de cadenas, en una lista ordenada alfabéticamente, de forma que la nueva cadena respete la ordenación. En el capítulo siguiente trataremos todo lo relativo a la ordenación.

Búsqueda de elementos alfanuméricos en listas y tablas

En el apartado tercero de este capítulo vimos cómo encontrar el mayor o menor de una lista, tanto numérica como alfanumérica. Ahora bien, la búsqueda de textos debe enfocarse en un contexto más amplio que el numérico. En este sentido y como complemento, estudiaremos a continuación la búsqueda de un elemento por igualdad o por contenido.

Búsqueda por igualdad

En una lista de nombres, por ejemplo, es frecuente tratar de buscar uno determinado y una vez localizado, listarlo junto con atributos inherentes al propio nombre. Veamos un ejemplo.

Imaginemos una breve agenda telefónica cuyo contenido sea: nombre, apellidos y teléfonos. En principio, cabría la posibilidad de utilizar 3 matrices de una dimensión, tales como:

N$ Para nombres
A$ Para apellidos
T$ para teléfonos

La N$ sería la matriz principal sobre la que vamos a efectuar la búsqueda de nombres y A$ y T$ serían las matrices de atributos. A un determinado nombre, elemento N$(i), le corresponderían los atributos A$(i) y T$(i) para los apellidos y teléfono, respectivamente.

Si mantenemos esta estructura matricial, la rutina de búsqueda podría ser:

```
1000 ' RUTINA DE BUSQUEDA POR IGUALDAD
1010 INPUT "NOMBRE A BUSCAR ";NOM$
1020 FOR K=1 TO N
1030 IF N$(K)=NOM$ THEN 1100
1040 NEXT K
1050 PRINT "NO EXISTE ";NOM$
1060 GOTO 1130
1100 PRINT N$(K)
1110 PRINT A$(K)
1120 PRINT T$(K)
1130 RETURN
```

Para poder utilizarla debe elaborar una rutina de entrada de datos como tal descritas en apartados anteriores.

Mediante la línea 1010 introducimos el nombre que buscamos, teniendo en cuenta que dicho nombre debe introducirse tal y como figura en la lista, es decir, respetando el tipo de letra, mayúsculas y minúsculas, tal como se comentó con anterioridad.

La línea 1020 establece un bucle extendido a todos los elementos de la lista (N elementos).

En 1030 se efectúa la comparación con el símbolo de igualdad —título de este epígrafe. Debe tenerse en cuenta que la igualdad ha de ser absoluta, carácter a carácter. Si dicha igualdad se verifica es que se ha encontrado el elemento buscado, y por tanto, pasamos a la línea 1100 y siguientes, que imprimen el nombre, los apellidos y el teléfono, aprovechando el índice K del bucle, común a nombre y atributos.

Si se agotase el bucle, sería una señal inequívoca de que ha recorrido toda la lista sin encontrar el nombre buscado; por ello, la 1050 nos avisa de este hecho y produce un retorno al programa principal.

Otra forma de enfocar el problema, más correcta, es utilizar una sola matriz de tres dimensiones. Si llamamos N$ a esta matriz, los elementos de la fila iésima representarían a:

N$(i,1) Nombre
N$(i,2) Apellidos
N$(i,3) Teléfono

Suponiendo que —por mayor claridad— no utilizamos los elementos cero de la tabla, como hemos hecho en muchos ejemplos anteriores e incluso en la rutina expuesta.

La confección de una rutina como la anterior, para este tipo de estructura matricial, no presenta ningún problema. Basémonos en la anterior.

```
1000 ' RUTINA DE BUSQUEDA POR IGUALDAD
1010 INPUT "NOMBRE A BUSCAR ";NOM$
1020 FOR K=1 TO N
1030 IF N$(K,1)=NOM$ THEN 1100
1040 NEXT K
1050 PRINT "NO EXISTE ";NOM$
1060 GOTO 1130
1100 PRINT N$(K,1)
1110 PRINT N$(K,2)
1120 PRINT N$(K,3)
1130 RETURN
```

El sistema expuesto a través de los elementos de búsqueda por igualdad, presenta varios problemas, entre ellos dos muy importantes. Por un lado, la entrada de datos para cargar las matrices ha de ser meticulosa, no vale introducir por ejemplo: PEPE y luego Antonio, hay que tomar un criterio fijo y, además, hay que tener cuidado con los espacios; a fin de cuentas, también son caracteres. Las funciones UPPER$ y LOWER$ nos serán útiles a estos efectos.

Si no conocemos con seguridad el tipo de letra con que se almacenaron los datos, puede procederse —antes de la comparación—, a homogeneizar el tipo de caracteres, tanto de la cadena buscada como de los elementos de la tabla, por ejemplo en el ejemplo anterior sería mejor cambiar la línea 1030 por

```
1030 IF UPPER$ (N$(K,1)) = UPPER$ (NOM$)
```

De esta forma las cadenas N$(K,1) y NOM$ se han convertido a mayúsculas, con independencia del tipo con que se escribieron anteriormente, sin embargo, las cadenas no han sido alteradas y seguirán estando formadas por los mismos tipos de caracteres, es decir, solo la comparación se realiza en mayúsculas. De igual forma podía utilizarse la función LOWER$, que pasa una cadena a minúsculas.

Por otro lado, lo más lógico en el ejemplo expuesto es introducir el nombre seguido de los apellidos, o apellidos seguidos del nombre, según los gustos o el destino del programa general. En este caso, para buscar un elemento, debemos recordar su nombre y dos apellidos escritos correctamente, cosa ya algo difícil, sobre todo si se trata de una agenda de regulares dimensiones.

Algunos de estos inconvenientes pueden solucionarse utilizando la técnica de búsqueda por inclusión, que exponemos a continuación.

Búsqueda por inclusión

El Mallard BASIC del Amstrad dispone de una función de búsqueda dentro de cadenas que nos facilitará el trabajo, aunque puede realizarse sin ella.

Esta función es INSTR que lleva 3 parámetros; el primero de ellos opcional.

INSTR (partida, cadena en la que busca, cadena buscada)

El primer parámetro indica la posición a partir de la cual se va a efectuar la búsqueda. Si no se le indica, toma 1 por defecto.

El segundo, la cadena en la cual va a buscar.

El tercero, la subcadena buscada.

Si la subcadena buscada existe, devuelve la posición del carácter a partir del cual se ha encontrado la coincidencia. Si no existe, devuelve un cero (0).

Veamos un ejemplo. Sea la cadena B$ = "Las noches de invierno son frías", en la cual se busca la subcadena A$ = "invierno".

```
INSTR (1,B$,A$) ó INSTR (B$,A$)
```

Nos devolverá el valor 15, que es la posición ocupada por la "i".

La aplicación en una búsqueda se realizará con un IF, comprobando que INSTR da un valor distinto de cero, lo cual indica que la subcadena está contenida.

Pruebe el ejemplo expuesto, tecleando las siguientes instrucciones sin número de línea:

```
B$ = "Las noches de invierno son frías"
A$ = "invierno"
PRINT INSTR (B$,A$)
```

Cambie ahora A$ por otro texto y compruebe los resultados. Le sugerimos los siguientes valores o contenidos de A$: "L", "de", "no", "frías". Como verá, localiza cualquier subcadena, sea o no palabra propiamente dicha.

Aún cuando lo normal será utilizar variables alfanuméricas con la instrucción INSTR, puede perfectamente utilizarse cadenas entrecomilladas, pruebe por ejemplo PRINT INSTR ("COCODRILO", "L") y obtendrá el valor 8.

Basándose en las facilidades que aporta el uso de esta función, puede elaborarse un ejemplo más completo y elástico en su utilización.

La matriz base de datos se establece en la siguiente forma

1) Nombre y apellidos	N$(i, 1)
2) Dirección	N$(i, 2)
3) Población	N$(i, 3)
4) Provincia	N$(i, 4)
5) Teléfono	N$(i, 5)

lo que implica dimensionar una matriz de dos dimensiones, la primera el total de nombres de la agenda y la segunda será cinco (5), correspondiente a los epígrafes anteriores.

DIM N$(N, 5) con OPTION BASE 1

El ejemplo consta de 3 partes

— Programa principal (10-100)

Inicializa la variable puntero que irá apuntando sucesivamente a cada uno de los nombres, a fin de llevar un control sobre los que se va realizando la búsqueda. Pide el nombre a buscar, que puede ser un nombre completo, un apellido, una inicial, etc.

Se ha previsto el caso en que pueden existir dos o más nombres que correspondan al buscado, a fin de que se muestren uno a uno los que van apareciendo y continue la búsqueda hasta agotar los datos (puntero > = N).

— Carga de datos en la matriz (110-240)

Los datos se han incluido en líneas DATA, una para cada nombre con sus atributos separados por comas. Por brevedad solo se incluyen tres, pero puede Vd. aumentarlos variando consecutivamente el valor de N en la línea 150.

La 170 dimensiona convenientemente la matriz en función de N y los bucles 180 y 190 controlan la lectura de datos.

La línea 240 produce el retorno al programa principal

— Rutina de búsqueda por inclusión (250-360)

La instrucción INSTR de la línea 270 incluida dentro de un bucle permite buscar la cadena NOM$ dentro de las cadenas de la matriz de nombres N$(K, 1).

La línea 290 controla el hecho de que el bucle se agote y como consecuencia no hay más datos.

Las líneas 310-350 es una pequeña rutina que imprime el nombre que corresponde al buscado, junto con sus atributos.

```
10 ´ PROGRAMA PRINCIPAL
20 PUNTERO=1
30 GOSUB 110: ´ Carga de datos en la matriz
40 INPUT "Nombre a buscar ";NOM$
50 GOSUB 250: ´ Rutina de busqueda
60 INPUT "desea continuar la busqueda (S/N) ";Z$
70 z$=UPPER$(Z$)
80 IF Z$="S" AND K<N THEN PUNTERO=K+1:GOSUB 250:GOTO 60
90 IF Z$="S" AND K>=N THEN PRINT"No hay mas nombres"
100 END
110 ´ CARGA DE DATOS EN LA MATRIZ
120 DATA PEPE GOTERA,OCA 24,MADRID,MADRID,6445566
130 DATA BENITO CARA DURA,RAMBLA 12,ALTILLO,GRANADA,2222222
140 DATA CARPANTA COMILON,PUENTE BAJO SN,ALPEDRETE,MADRID,0
150 N=3: ´Numero de datos
160 OPTION BASE 1
170 DIM N$(N,5)
180 FOR K=PUNTERO TO N
190 FOR I=1 TO 5
200 READ N$(K,I)
210 NEXT I
220 NEXT K
230 PRINT"Datos cargados"
240 RETURN
250 ´ RUTINA DE BUSQUEDA POR INCLUSION
260 FOR K=PUNTERO TO N
270 IF INSTR (N$(K,1),NOM$)<>0 THEN 310
280 NEXT K
290 PRINT"FIN DE LA LISTA DE NOMBRES"
300 GOTO 100
310 PRINT"*****************************************"
320 FOR I=1 TO 5
330 PRINT N$(K,I)
340 NEXT I
350 PRINT"*****************************************"
360 RETURN
```

Aunque de esta forma puede construirse una agenda, en los capítulos dedicados a los ficheros se incluyen ejemplo completos de agendas mucho más eficaces.

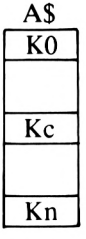
En el capítulo 5, puede ver un programa (BIBLIO) que permite llevar una biblioteca, en el cual se utiliza la técnica de búsqueda por inclusión.

Búsqueda en listas ordenadas

En los dos apartados anteriores nos hemos referido, aunque de forma implícita, a listas no ordenadas, donde la localización de un elemento ha de realizarse, "mirando" desde el primero al último, pues no existe ningún otro criterio de búsqueda.

Ahora bien, si una lista está ordenada alfabéticamente y el elemento buscado comienza por z, evidentemente se encontrará por el final. Luego en listas ordenadas, podemos establecer alguna regla o algoritmo de búsqueda, conducente a conseguir una mayor velocidad en la misma.

Uno de estos métodos es el de BUSQUEDA BINARIA. Su fundamento resulta sencillo: si comparamos el elemento buscado con el central de la lista y resulta menor, es que estará por encima, por el contrario, estará por debajo. Veámoslo gráficamente:



= índice del primer elemento

= índice del elemento central

= índice del último elemento

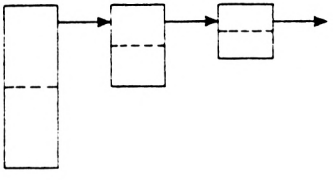
Si la cadena buscada es B$ y resulta

B$ > A$(Kc) —si→ B$, está en la primera mitad

↓ no

B$, está en la segunda mitad

Con una sola partición podemos desechar la mitad de los elementos, de manera que si realizamos una nueva partición, comparando con el elemento central de la mitad válida, podemos quedarnos con un cuarto de los elementos para seguir la búsqueda. De esta forma la convergencia es muy rápida.



elemento
buscado

Búsqueda binaria
Particiones sucesivas a la mitad

Basándonos en este sencillo método, podemos elaborar una rutina rápida de búsqueda de cadenas alfanuméricas en listas ordenadas. Ha de tenerse en cuenta que ca-

da partición supone calcular los nuevos límites de la lista y el elemento central, que corresponde al valor medio de los límites. Dichos elementos singulares vienen representados por las variables K0, Kn y Kc, respectivamente.

```
10 ' CARGA DE LA LISTA ORDENADA
20 N=20 DIM A$(N)
30 FOR K=1 TO N
40 READ A$(K)
50 NEXT K
60 DATA AREA,AXILA,BALA,BARCO,CEBRA,CORCHO,DATIL,DEDO,DINAMO,DOLOR
70 DATA FELIZ,FUGA,GATO,HUECO,INDIO,JAMON,KILO,LIMON,MANGA,PERRO
80   BUSQUEDA BINARIA
90 INPUT "NOMBRE BUSCADO";NOM$
100 K0=1 KN=N
110 KC=INT((KN-K0)/2)+K0
120 IF NOM$=A$(KC) THEN 160
130 IF NOM$>A$(KC) THEN K0=KC·GOTO 110
140 KN=KC
150 GOTO 110
160 PRINT "El nombre ",NOM$;" ocupa el lugar ";KC
```

Las líneas 10 a 70 son auxiliares y su misión es comprobar el funcionamiento de la rutina.

Como origen, se toma en la línea 100 los valores 1 y N para los límites; el valor medio correspondiente se calcula en 110.

La línea 120 controla la búsqueda por igualdad, aunque se podría haber adoptado el método de la inclusión.

La línea 130 sirve de báscula para la partición. Observe cómo el nuevo valor de K0, en el caso de que el nombre buscado esté por debajo, se hace igual al valor central y en caso contrario, el límite inferior Kn, se hace igual al central.

Pruebe esta rutina y comprobará su perfecto funcionamiento, pero introduzca un nombre no contenido en las DATA. Ahora el programa buscará sin descanso y sin detenerse y, evidentemente, no la encontrará, así que debemos añadir un control para poder salir de este bucle infinito.

¿Qué condición pondremos? Introduzca la siguiente línea para ver qué pasa con los valores de límites y centro:

```
115 PRINT K0,KC,KN
```

Haga un nuevo RUN, introduzca palabras no existentes en las DATAS y observará que en la quinta o sexta vuelta, los 3 límites son iguales. Como el número de vuelta depende evidentemente del número de la palabras de la lista, no nos sirve como indicador, así que tomamos la observación de las igualdades como conclusión, bastándonos estrictamente que sean iguales los dos primeros.

Añadamos las siguientes líneas y ahora sí funcionará con cualquier palabra, exista o no en la lista.

```
115 IF K0 = KC THEN 180
```

```
170 GOTO 190
180 PRINT "No existe", NOM$
190 END
```

7. EJERCICIOS RESUELTOS SOBRE MATRICES

Ejercicio 1: Dadas las calificaciones obtenidas por los alumnos de una clase, por asignatura, obtener la nota media de cada alumno, la nota media de cada asignatura, el número de alumnos aprobados y suspensos y el porcentaje de ambos.

Datos: Vamos a utilizar como ejemplo sólo 5 alumnos, aunque el programa es válido para cualquier número de alumnos. Asimismo, supondremos 6 asignaturas, valor que puede también variarse.

	ASIGNATURA					
Nombre del Alumno	*Mat*	*Soc*	*Dib*	*Len*	*Rel*	*Quim*
Antonio García	6	7	8	7	9	9
Luis Conde	8	9	10	9	10	9
José Moreno	4	5	3	5	5	2
Rubén López	3	2	6	5	5	4
Luis Díaz	7	6	5	4	6	5

Dadas las premisas del ejercicio, utilizaremos 3 matrices

CONTENIDO	*DIMENSION*	*TIPO*
Nombre de los alumnos	(5)	alfanumérica
Asignaturas	(6)	alfanumérica
Notas	(5,6)	numérica

La nota media de cada alumno será la suma de los elementos de la matriz por filas, dividido por 6; mientras que la media de cada asignatura será la suma por columnas, dividido por 5 (alumnos).

La operación de sumar filas y columnas de una tabla fue explicada en el segundo apartado de este capítulo, en cuyo ejercicio nos apoyaremos.

Los datos de las matrices se cargarán desde el teclado, por medio de sentencias INPUT. Más adelante veremos la forma de almacenar los datos en ficheros.

```
10 ' N O T A S    M E D I A S
20 OPTION BASE 1
30 INPUT "NUMERO DE ALUMNOS";AL
40 INPUT "NUMERO DE ASIGNATURAS";AS
50 'DIMENSIONAMIENTO Y ENTRADA DE DATOS
60 DIM AL$(AL),AS$(AS),N(AL,AS)
70 FOR K=1 TO AL
80 PRINT "NOMBRE DEL ALUMNO",K
90 INPUT AL$(K)
100 NEXT K
110 FOR K=1 TO AS
120 PRINT "ASIGNATURA";K
130 INPUT AS$(K)
140 NEXT K
150 PRINT CHR$(27)+"F"+CHR$(27)+"H"
160 FOR F=1 TO AL
170 PRINT "--------------------------------------"
180 PRINT "NOTAS DEL ALUMNO: ";AL$(F)
190 FOR C=1 TO AS
200 PRINT AS$(C);
210 INPUT N(F,C)
220 NEXT C,F
230 ' CALCULO DE LAS MEDIAS
235 ' Por alumno
236 PRINT "NOTAS MEDIAS DE LOS ALUMNOS"
237 Z$=STRING$(40,"=")·PRINT Z$
250 FOR F=1 TO AL
260 FOR C=1 TO AS
270 N(F,0)=N(F,0)+N(F,C)
280 NEXT C
290 PRINT "NOTA MEDIA DE ";AL$(F);"  "; USING "## #";N(F,0)/AS
300 NEXT F
310 ' Por asignatura
315 PRINT :PRINT "NOTAS MEDIAS DE LAS ASIGNATURAS"
316 PRINT Z$
320 FOR C=1 TO AS
330 FOR F=1 TO AL
340 N(0,C)=N(0,C)+N(F,C)
350 NEXT F
360 PRINT "NOTA MEDIA DE ",AS$(C);"  "; USING "## #",N(0,C)/AL
370 NEXT C
400 ' ALUMNOS APROBADOS Y SUSPENDIDOS
410 FOR F=1 TO AL
420 FOR C=1 TO AS
430 IF N(F,C)<5 THEN SUS=1
440 NEXT C
450 SUSPENSOS=SUSPENSOS+SUS
460 NEXT F
470 APROBADOS=AL-SUSPENSOS
480 PRINT :PRINT "APROBRADOS=";APROBADOS,"%="; USING "#### ##";APROBADOS/AL*100
490 PRINT :PRINT "SUSPENSOS =";SUSPENSOS,"%="; USING "#### ##";SUSPENSOS/AL*100
```

Ejercicio 2: Hallar el producto de una matriz por un escalar. El producto de una matriz (A) por un escalar K, es otra matriz (B), tal que todos sus elementos se obtienen B(i,j) = A(i,j)*K.

A continuación exponemos la rutina para calcular la matriz (B). La entrada y salida de resultados la dejaremos a criterio del lector. En los ejercicios anteriores hay varios ejemplos que sin duda le ayudarán.

```
10 ' PRODUCTO DE UNA MATRIZ POR UN ESCALAR
20 ' AGREGAR LA RUTINA DE CARGA
30 '              MATRIZ A(M,N)
40 '              ESCALAR=K
```

```
50 '
200 ' MATRIZ PRODUCTO
210 FOR F=1 TO M
220 FOR C=1 TO N
230 B(F,C)=A(F,C)*K
240 NEXT C,F
300 ' AGREGAR RUTINA DE SALIDA DE RESULTADOS
310 '
```

Ejercicio 3: Hallar la inversa de una matriz.

Este ejercicio y los dos siguientes van destinados al profesional y estudiante que utiliza y realiza ejercicios en los que es necesario el cálculo de la matriz inversa. Por todo ello, pasamos a escribir el programa, remitiendo a un libro de Matemáticas a aquellos lectores interesados en el proceso de cálculo.

Como comprobación del programa, puede utilizar los siguientes ejemplos:

$$A = \begin{pmatrix} -1 & 6 \\ 4 & 3 \end{pmatrix} \quad \text{Inversa } B = \begin{pmatrix} -0.1111 & 0.222 \\ 0.1481 & 0.037 \end{pmatrix} \qquad B = A^{-1}$$

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad \text{No tiene inversa (matriz singular)}$$

Nota: Si desea obtener más decimales, cambie el PRINT USING de la línea 560.

```
10 ' CALCULO DE LA MATRIZ INVERSA
20 INPUT "ORDEN DE LA MATRIZ ";N
30 OPTION BASE 1
40 DIM A(N,N),B(N,N)
50 ' Entrada de datos de la matriz A
60 FOR F=1 TO N
70 PRINT "F I L A ";F
80 FOR C=1 TO N
90 PRINT "COLUMNA ";C;
100 INPUT A(F,C)
110 NEXT C
120 PRINT "----------------------------------------"
130 NEXT F
140 ' Validez de (A)
150 FOR F=1 TO N
160 B(F,F)=1
170 NEXT F
180 FOR C=1 TO N
190 FOR F=C TO N
200 IF A(F,C)<>0 THEN 230
210 NEXT F
220 PRINT "LA MATRIZ (A) ES SINGULAR Y POR TANTO NO TIENE INVERSA" END
230 ' Calculo de la matriz inversa (B)
240 FOR k=1 TO n
250 AUX=A(C,K)
260 A(C,K)=A(F,K)
270 A(F,K)=AUX
280 AUXB=B(C,K)
290 B(F,K)=AUXB
300 NEXT K
310 Z=1/A(C,C)
320 FOR K=1 TO N
```

```
330 A(C,K)=A(C,K)*Z
340 B(C,K)=B(C,K)*Z
350 NEXT K
360 FOR I=1 TO N
370 IF I=C THEN 430
380 Z=-A(I,C)
390 FOR J=1 TO N
400 A(I,J)=A(I,J)+A(C,J)*Z
410 B(I,J)=B(I,J)+B(C,J)*Z
420 NEXT J
430 NEXT I
440 NEXT C
450 ' Salida de resultados
460 PRINT "M A T R I Z     I N V E R S A"
470 FOR F=1 TO N
480 FOR C=1 TO N
490 PRINT  USING "##### ####",B(F,C),
500 NEXT C
510 PRINT
520 NEXT F
```

Ejercicio 4: Determinante de una matriz cuadrada.

Este programa permite calcular el valor del determinante de una matriz cuadrada (número de filas = número de columnas).

El procedimiento de cálculo utilizado es el de triangulación de la matriz. De este método resulta que el producto de los elementos de la diagonal de la matriz triangulada es el valor del determinante.

Si se desea, puede incorporarse a la entrada de datos una rutina de depuración. Para ello, puede adaptarse la rutina M7 (líneas 200 a 400). Los valores numéricos 5 y 7 de dicha rutina deben eliminarse y dejarlos como N (orden del determinante).

```
10 ' RESOLUCION DE SISTEMAS DE ECUACIONES
20 OPTION BASE 1
30  PRINT "Este programa permite calcular un sistema de  <n> ecuaciones con <n>
incognitas de la forma"
40 PRINT " a11  X1 + a12  X2 + a13 X3 + a1n Xn = b1  "
50 PRINT " a21  X1 + a22  X2 + a23 X3 + a2n Xn = b2  "
60 PRINT "                                           "
70 PRINT " an1  X1 + an2  X2 + an3 X3 + ann Xn = bn  "
80 LOCATE 12,20:INPUT "INTRODUZCA EL NUMERO DE ECUACIONES";EC
90 DIM A(EC,EC+1)
100 ' Entrada de datos
110 FOR F=1 TO EC
120 PRINT "INTRODUZCA LOS COEFICIENTES DE LA ECUACION";F
130 FOR C=1 TO EC
140  PRINT "A";F;C;"= "·LOCATE 10,C+1:INPUT "",A(F,C)
150 NEXT C
160 INPUT "TERMINO INDEPENDIENTE ";A(F,EC+1)
170 NEXT F
180 'sistema en pantalla
190 FOR F=1 TO EC
200 FOR c=1 TO EC
210 PRINT A(F,C),"  ",
220 NEXT C
230 PRINT "·";A(F,EC+1)
240 NEXT F
250  PRINT :INPUT "DATOS CORRECTOS (S/N)     ",Z$
260 IF Z$="S" OR Z$="s" THEN 320
270 INPUT "CUANTOS DATOS INCORRECTOS",IN
280 FOR K= 1 TO IN
290 INPUT "INTRODUZCIR Fila,Columna,Valor correcto',F,C,A(F,C)
```

```
300 NEXT K
310 GOTO 180
320 ' Calculo
330 FOR F=1 TO EC
340 FOR C=F TO EC
350 IF A(F,C)<>0 THEN 370 ELSE NEXT C
360 PRINT "   EL SISTEMA TIENE INFINITAS SOLUCIONES    " END
370 FOR K=1 TO EC+1
380 X=A(F,K)·A(F,K)=A(C,K) A(C,K)=X
390 NEXT K
400 Y=1/A(F,F)
410 FOR K=1 TO EC+1·A(F,K)=A(F,K)*Y  NEXT K
420 FOR C=1 TO EC
430 IF C=F THEN 480
440 Y=-A(C,F)
450 FOR K=1 TO EC+1
460 A(C,K)=A(C,K)+A(F,K)*Y
470 NEXT K
480 NEXT C
490 NEXT F
500 'Salida de resultados
510 PRINT "   S O L U C I O N    D E L    S I S T E M A"
520      PRINT "=============================================="
530 FOR C=1 TO EC
540 PRINT "X(";C,") =";  USING "##### ###",A(C,EC+1)
550 NEXT C
```

Ejercicio 5: Resolución de sistemas de ecuaciones lineales.

El programa que acompañamos permite obtener las soluciones de un sistema de ecuaciones lineales con el mismo número de ecuaciones que incógnitas, siempre que el determinante de la matriz de coeficientes no sea nulo.

Se ha incluido una rutina de depuración de los datos de entrada, similar a la expuesta en el apartado correspondiente.

Si se desea, pueden eliminarse las líneas 40 a 80, que son exclusivamente de presentación de la forma general de un sistema de ecuaciones lineales.

```
10   CALCULO DEL DETERMINANTE DE UNA MATRIZ CUADRADA
20 INPUT 'ORDEN DEL DETERMINANTE ",N
30 OPTION BASE 1
40 DIM A(N,N),Z(N)
50 ' Entrada de datos de la matriz
60 FOR F=1 TO N
70 PRINT "F I L A ";F
80 FOR C=1 TO N
90 PRINT "COLUMNA ';C;
100 INPUT A(F,C)
110 NEXT C
120 PRINT "-------------------------------------
130 NEXT F
140 ' Validez del determinante (matriz no singular)
150 FOR C=1 TO N
160 FOR F=C TO N
170 IF A(F,C)<>0 THEN 200
180 NEXT F
190 PRINT "LA MATRIZ  ES SINGULAR Y POR TANTO EL DETERMINANTE NULO" END
200 ' Calculo de la matriz inversa (B)
210 FOR k=1 TO n
220 AUX=A(C K)
230 A(C K)=A(F K)
240 A(F,K)=AUX
250 NEXT K
```

```
260 Z(C)=A(C,C)
270 Z=1/A(C,C)
280 FOR K=1 TO N
290 A(C,K)=A(C,K)*Z
300 NEXT K
310 FOR I=1 TO N
320 IF I=C THEN 370
330 Z=-A(I,C)
340 FOR J=1 TO N
350 A(I,J)=A(I,J)+A(C,J)*Z
360 NEXT J
370 NEXT I
380 NEXT C
390 ' Calculo de los productos de la diagonal
400 P=1
410 FOR K=1 TO N P=P*Z(K)·NEXT K
420 ' Salida de resultados
430 PRINT :PRINT "El valor del determinante es  ,P
```

CAPITULO 3

ORDENACION

1. INTRODUCCION
2. ORDENACION POR INTERCAMBIO
3. METODO BUBBLE SORT O DE LA BURBUJA
4. BUBBLE SORT CON MARCA
5. INTERCAMBIO RETARDADO
6. BUSQUEDA DEL MINIMO Y MAXIMO. DOBLE PASADA
7. METODO DE SHELL-METZNER
8. QUICKSORT
9. PROGRAMA GENERAL DE ORDENACION
10. PROGRAMA ORDENACION DE FECHAS

3. Ordenación

1. INTRODUCCION

Dentro de la gestión de datos, la obtención de listas ordenadas de los mismos, ocupan un lugar clave. Por ello, los programas de tratamiento de datos incorporan una o varias rutinas de clasificación, tanto para datos numéricos como alfanuméricos.

Las rutinas de clasificación u ordenación, conocidas como SORTs son, por tanto, independientes de los propios programas a que acompañan y secundarias, en el sentido de que el tiempo invertido por ellas debe ser mínimo, a fin de ceder el control lo más rápidamente al programa principal.

Los tipos de ordenación posibles dependen de la naturaleza de los datos y en términos generales pueden clasificarse de la forma siguiente:

Datos numéricos	De menor a mayor
	De mayor a menor
Datos alfanuméricos	Alfabéticamente

Por supuesto que existen otros tipos de ordenación: por grupos alfabéticos (v.g. cadenas que comienzan por la misma letra); por marcas (v.g. números comprendidos entre 1.000 y 2.000); por atributos comunes (v.g. personas con la misma nacionalidad), etc.

Pero si variados son los tipos de ordenación que pueden establecerse, aún lo son más los métodos existentes para conseguirlo, desde los más sencillos e intuitivos, hasta sofisticados sistemas basados en algoritmos matemáticos complejos.

Esta diversidad de métodos no es de ningún modo caprichosa, sino que obedece a necesidades muy variadas. No es lo mismo tratar un pequeño número de datos que uno grande, ni tampoco resulta lo mismo ordenar una lista poco desordenada, que una de carácter caótico o aleatorio.

Por todo ello, no puede hablarse de un método mejor o peor que otro, si no va unido este juicio a dos parámetros: tamaño del grupo de datos y características del mismo.

En los epígrafes que siguen se exponen algunas de estas técnicas, indicándose la idoneidad de su aplicación según los parámetros citados.

Todas las rutinas de ordenación expuestas se complementan con las incluidas al final del capítulo y que se citan a continuación:

1ª Entrada de datos. Esta rutina permite la carga de datos que componen la lista, desde el teclado, tanto numéricas como alfanuméricas.

2ª Grabación de datos en fichero. Los datos introducidos por teclado son salvados en un fichero secuencial.

3ª Lectura de ficheros. permite recuperar datos archivados.

4ª Salida de datos ordenados: Presenta los datos una vez ordenados por cualquiera de los métodos que se exponen.

Estas cuatro rutinas, así como las de ordenación, se han elaborado de forma que sean compatibles entre sí, lo cual permite que sean ensambladas mediante MERGE, constituyendo un completo programa de ordenación.

PGM.ORDENACION = R.1 + R.2 + R.3 + R.ORDENACION + R.4

Para mayor comodidad del lector, se incluye un programa con las cuatro rutinas ensambladas y dirigidas desde un menú que permite además la carga (MERGE) en la rutina de ordenación deseada, trabajando todo conjuntamente.

2. ORDENACION POR INTERCAMBIO

Este método de ordenación, también conocido como dicotómico, es quizá el más intuitivo. Se basa en recorrer la lista de datos desde el principio, comparando elementos contiguos e intercambiándolos si no están ordenados.

En una lista genérica A$(), se compara A$(1) con A$(2) y se coloca como A$(2) el mayor de los dos, luego se compara A$(2) con A$(3) y así sucesivamente.

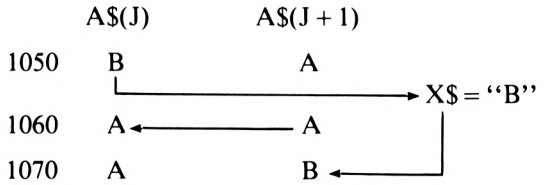
Cuando se termina de recorrer la lista, tendremos el mayor situado en último lugar, es decir, en su sitio. Por tanto, en la segunda vuelta (2° bucle) ya no es necesario llegar hasta él, sino hasta el antepenúltimo y en general hasta el anterior al último elemento ordenado. Es decir, si I es la variable del primer bucle, el límite del segundo será N—I, siendo N el número de datos a ordenar.

Veamos cómo será una rutina que utilice este método.

```
1000 ' ORDENACION POR INTERCAMBIO
1015 ' ..........ORDENA1 ........
1020 FOR I=1 TO N-1
1030 FOR J=1 TO N-I
1040 IF A$(J)<=A$(J+1) THEN 1080
1050 X$=A$(J)
1060 A$(J)=A$(J+1)
1070 A$(J+1)=X$
1080 NEXT J
1090 NEXT I
```

El esquema de funcionamiento de la rutina es el siguiente:

Suponiendo que el elemento A$(J) = "B" y A$(J + 1) = "A"



El método resulta sencillo y es quizá su mayor virtud; en cambio, su velocidad resulta baja. Hay que tener en cuenta que el número de vueltas es siempre fijo, de manera que si una lista está casi ordenada, tardará lo mismo que en otra aleatoria.

Otro tanto a su favor, aunque es común a otros métodos, es que no consume memoria, puesto que únicamente se utiliza una variable, la auxiliar (X$), dado que la ordenación se realiza sobre la propia lista.

3. METODO DEL BUBBLE SORT O DE LA BURBUJA

Las comparaciones se realizan comenzando por el primer elemento frente a todos los demás. Si en una comparación resulta ser menor, conserva su sitio, en caso contrario se intercambia.

En cada pasada el menor queda, pues, situado arriba, por ello recibe el nombre de método de burbuja, ya que los números ascienden por la lista como las burbujas en una bebida gaseosa. Por cierto, a estas alturas de la lectura del libro, quizá sea el momento de prepararnos un refresco a la vez que utilizamos el bubble sort.

Como puede observarse, en cada pasada se está seleccionando el mínimo de la serie, por lo que también se le conoce como método de selección del mínimo. Evidentemente, si se desea una ordenación en sentido inverso, el método es válido, basta con selecciona el máximo en vez del mínimo, ello equivale a sustituir el operador de relación "> " de la línea 1040 por el "< ", en la rutina que incluimos a continuación.

```
1000 ' ORDENACION METODO DE LA BURBUJA
1015 ' .......... ORDENA2 ............
1020 FOR I=1 TO N-1
1030 FOR J=I+1 TO N
1040 IF A$(I)>A$(J) THEN X$=A$(I):A$(I)=
A$(J):A$(J)=X$
1050 NEXT J
1060 NEXT I
```

Este método es muy similar al anterior. Sólo utiliza una variable auxiliar, de forma que los intercambios se realizan dentro de la propia matriz, por lo que el consumo de memoria es mínimo.

4. BUBBLE SORT CON MARCA

Se trata de una variante del método anterior, que introduce una mejora en el tiempo de procesado.

En el anterior, si la lista a tratar está compuesta de elementos "cuasi ordenados", la ordenación debiera ser más rápida, pero el número de pasadas es fijo, completamente independiente del grado de orden o anarquía en que se encuentra la lista inicial.

Si tras una pasada ninguno de los elementos de la serie se ha movido, quiere decirse que la lista ya está ordenada, por lo que lo lógico es salir de los bucles, abandonando el proceso.

Este método incorpora una marca, consistente en hacer bascular el valor de una variable entre 0 y 1. Si hay cambio, toma el valor 0, si no lo hay toma el 1 (por supuesto que pueden tomarse dichos valores al revés, o de cualquier otra forma similar). El control de la marca puede realizarse mediante IF o bien aprovechando la potencia de la sentencia WHILE, como en el listado que se ofrece a continuación.

```
1000 ' ORDENACION. BURBUJA CON MARCA
1015 ' .......... ORDENA3 ..............
1020 MARCA=0
1030 WHILE MARCA=0
1040 MARCA=1
1050 FOR I=1 TO N-1
1060 IF A$(I)>A$(I+1) THEN X$=A$(I):A$(I
)=A$(I+1):A$(I+1)=X$:MARCA=0
1070 NEXT I
1080 WEND
```

Este método, habida cuenta de sus características, está especialmente indicado en listas con datos casi ordenados, como por ejemplo listas ordenadas, en las que se intercalan o sustituyen nuevos datos y deben volver a ordenarse (ficheros dinámicos).

5. INTERCAMBIO RETARDADO

Una mejora de todos los métodos basados en la búsqueda del mínimo por intercambios sucesivos, consiste en retardar dichos intercambios, en tanto en cuanto que dicho mínimo no lo sea de toda la lista (mínimo absoluto).

Es decir, en cada pasada se extrae el mínimo absoluto. Una vez encontrado se sitúa en su sitio. De esta forma en la comparación de dos elementos no se produce intercambio, hasta haber encontrado dicho mínimo (retardo en el intercambio).

El proceso se repite nuevamente para encontrar el segundo mínimo, y así sucesivamente, la siguiente rutina emplea dicho método.

```
1000 ' ORDENACION POR INTERCAMBIO RETARD
ADO
1015 ' ........... ORDENA4 ..............
1020 FOR I=1 TO N-1
1030 MINIMO=I
1040 FOR J=I+1 TO N
1050 ' Busqueda del minimo .....
1060 IF A$(J)<A$(MINIMO) THEN MINIMO=J
1070 NEXT J
1080 ' Intercambio .............
1090 IF MINIMO-I        THEN X$=A$(MINIMO
):A$(MINIMO)=A$(I):A$(I)=X$
1100 NEXT I
```

El bucle interno (líneas 1040-1070) es el encargado de encontrar el mínimo en cada pasada. Tras salir de este bucle, se produce la colocación del mínimo encontrado en el lugar que le corresponde. Observe que en la búsqueda del mínimo, lo que se recoge no es su valor, sino simplemente el lugar que ocupa.

6. BUSQUEDA DEL MINIMO Y MAXIMO. DOBLE PASADA

Si en el método anterior se buscaba el mínimo, parece lógico buscar simultáneamente el máximo. El mínimo irá al primer lugar y el máximo al último, por lo que la siguiente pasada debe realizarse sobre N—2 elementos, en vez de sobre N—1. En pasadas sucesivas, los límites del bucle serán cada vez menores, contando de dos en dos, todo lo cual da más velocidad al proceso.

Compare ambas rutinas y haga una prueba de velocidad, con un número de datos suficientemente grande.

```
1000 ´ ORDENACION METODO DOBLE PASADA
1015 ´ ..........  ORDENA5 ..........
1020 PRIMERO=1:ULTIMO=N-1
1030 WHILE ULTIMO>PRIMERO
1040 FOR K=1 TO N-1
1050 IF A$(K)>A$(K+1) THEN X$=A$(K):
A$(K)=A$(K+1):A$(K+1)=X$:Y=K
1060 NEXT K
1070 ULTIMO=Y-1
1080 FOR K=ULTIMO TO PRIMERO STEP -1
1090 IF A$(K)>A$(K+1) THEN X$=A$(K):
A$(K)=A$(K+1)·A$(K+1)=X$:Y=K
1100 NEXT K
1110 PRIMERO=Y+1
1120 WEND
```

7. METODO DE SHELL-METZNER

Todos los métodos expertos tienen el inconveniente común de su lentitud (en mayor o menor grado) cuando el número de datos es grande y se encuentran muy desordenados, debido a que el número de vueltas de los bucles resulta excesivamente elevado.

El Shell-sort —o método de "concha"— está basado en un algoritmo matemático, algo más complicado que la simple comparación e intercambio, ya que en él la comparación no se realiza como parece más lógico —entre elementos consecutivos— sino entre elementos separados entre sí en una distancia constante, que llamaremos CTE, cuyo valor más habitual es:

$$CTE = (2\uparrow X-1)/2$$

Donde X es el exponente de la menor de las potencias de 2, de forma que $2\uparrow X > N$ (número de elementos de la lista).

De esta forma, las comparaciones se harán entre

A$(K) y A$(K + CTE)

donde K representa la variable del bucle.

Aunque existen otros algoritmos para calcular CTE, el indicado es uno de los más utilizados, pues produce muy buenos resultados.

Veamos el listado:

```
1000 ' ORDENACION METODO SHELL (1)
1015 ' ........... ORDENA6 ........
1020 X=FIX (LOG(N)/LOG(2))
1030 CTE=(2↑X-1)/2
1040 WHILE CTE
1050 X=N-CTE:Y=1
1060 WHILE Z<=X
1070 K=Z
1080 IF A$(K)>A$(K+CTE) THEN X$=A$(K):
A$(K)=A$(K+CTE):A$(K+CTE)=x$:K=K-CTE:
IF K>1 THEN 1100
```

Una variante también muy utilizada es hacer que CTE valga

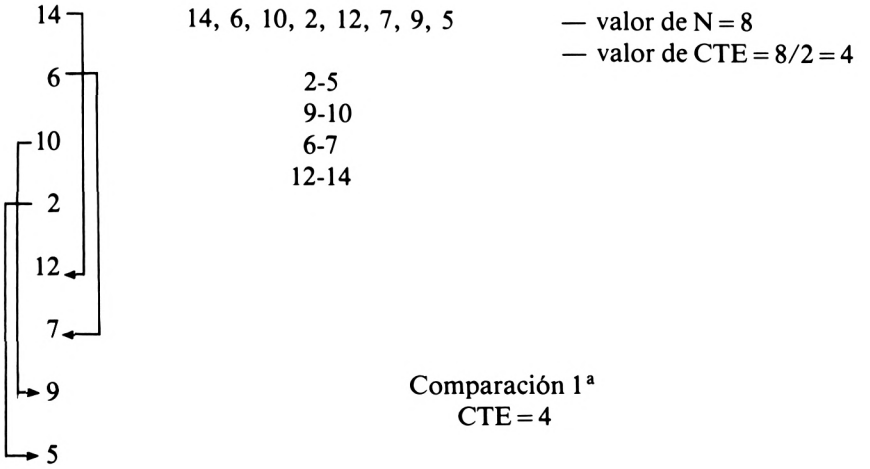
$$CTE = N/2$$

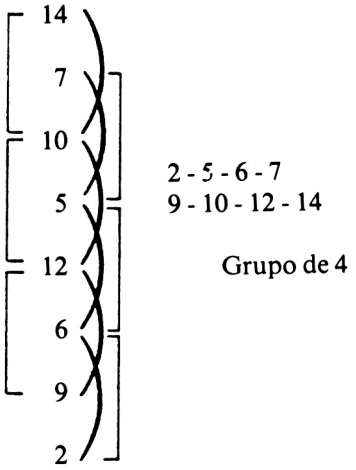
de forma que las comparaciones se realizarán

$$A\$(K) \text{ y } A\$(K + N/2)$$

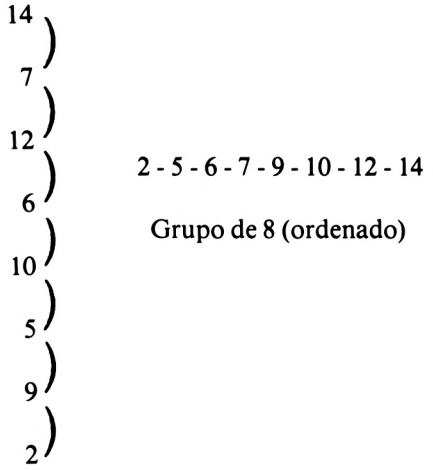
El funcionamiento es el siguiente: primero se comparan los elementos dos a dos en la forma indicada y se ordenan. Segundo, se fusionan los grupos de dos elementos en grupos de cuatro. Tercero, los grupos de cuatro en grupos de ocho y así hasta obtener la lista clasificada.

Como ejemplo gráfico, a fin de facilitar el seguimiento del método, veamos cómo se ordenaría la siguiente lista numérica:





Comparación 2ª



Comparación 3ª

A continuación exponemos el listado de esta variante del método de Shell.

```
1000 ´ ORDENACION. METODO DE SHELL (2)
1015 ´ ..... ... ORDENA7 ..........
1020 CTE=INT((N+1)/2)
1030 FOR K=1 TO N-CTE
1040 IF A$(K)<=A$(K+CTE) THEN 1060
1050 X$=A$(K):A$(K)=A$(K+CTE):A$(K+CTE)=X$
1060 NEXT K
1070 IF CTE=1 THEN 1100 ELSE CTE=CTE-1·GOTO 1030
1100 ´ SALIDA DE DATOS
```

8. QUICKSORT

El método de ordenación Quicksort, último de los expuestos, es sin duda el más eficaz en el tratamiento de grandes cantidades de datos y muy desordenados.

La metodología no es difícil, se trata de realizar unas peticiones en la lista, tratándolas por separado y con sucesivas subdivisiones. Se toma el primer elemento de la lista desordenada como pivote, se comparan con el resto de los elementos, colocándolos en dos grupos, los menores encima y los mayores debajo.

MENORES	ELEMENTO PRIMERO (PIVOTE)	MAYORES

A continuación se realiza la partición de la lista en dos y se vuelve a elegir un pivote por grupo, y por comparación se vuelve a obtener dos subconjuntos de cada uno de ellos y se repite la fragmentación y el proceso hasta que no pueden volverse a dividir.

Como crítica al método, debemos indicar que frente a una mayor optimización en el tratamiento de listas grandes y desordenadas, se encuentra el inconveniente de su mayor consumo de memoria al utilizar matrices auxiliares, aunque hoy día las memorias de los ordenadores han dejado de ser problema.

Veamos el listado de esta rutina.

```
1000 ' METODO DE ORDENACION QUICKSORT
1015 ' ........... ORDENA8 ............
1020 CON=FIX(LOG(N)/LOG(2))+4
1030 ERASE X:DIM X(CON,2)
1040  X(1,1)=1:X(1,2)=N
1050 WHILE CON
1070 C1=X(CON,1):F1=X(CON,2)
1080 CON=CON-1
1090 X$=A$(C1):C2=C1
1100 F2=F1:F1=F1+1
1110 F1=F1-1
1120 IF F1=C1 THEN 1170
1130 IF A$(F1)<X$ THEN A$(C1)=A$(F1) ELSE 1110
1140 C1=C1+1
1150 IF C1=F1 THEN 1170
1155 IF A$(C1)>X$ THEN A$(F1)=A$(C1):GOTO 1110 ELS
E 1140
1160 IF A$(C1)>X$ THEN A$(F1)=A$(C1):GOTO 1110
1170 A$(C1)=X$
1180 IF C2<C1-1 THEN X(CON+1,1)=C2:X(CON+1,2)=C1-1
:CON=CON+1
1190 IF F1+1<F2 THEN X(CON+1,1)=F1+1:X(CON+1,2)=F2
:CON=CON+1
1200 WEND
```

9. PROGRAMA GENERAL DE ORDENACION

Como ya se indicó en el epígrafe 1 de este capítulo, y con el fin de que el lector pueda probar (si lo desea) los distintos métodos de ordenación expuestos, se incluyen en este apéndice una colección de rutinas que permiten trabajar en este campo de la ordenación y con cualquiera de los métodos.

Las rutinas se han preparado de forma que esté asegurada su compatibilidad conjunta. Como referencia, incluimos a continuación su número de orden, nombre y números de línea que abarca cada una de ellas.

Rutina	*Nombre*	*Nº Línea*	*Comentarios*
1ª	MENU	60 - 210	Menú de opciones que controla funcionamiento conjunto
2ª	ENTRADAT	300 - 470	Entrada de datos (teclado)
3ª	GRAFICHE	500 - 610	Salva el fichero en disco/cinta
4ª	LEEFICHE	700 - 820	Lee el fichero de disco/cinta
5ª	ORDENAn	1000 - VAR	Rutina de ordenación nº n
6ª	SALDATOS	2000 - 2120	Salida de datos ordenados

A continuación se incluyen las citadas rutinas listadas y comentadas.

1ª MENU. Consiste en un menú directorio desde el cual puede accederse a resto de las rutinas. Contiene las siguientes opciones:

1. ENTRADA DE DATOS
2. GRABAR FICHERO
3. LEER FICHERO
4. ORDENACION
5. SALIDA DATOS ORDENADOS
6. CARGAR RUTINA DE ORDENACION

Las opciones 1 a 5, corresponden obviamente a las rutinas del cuadro anterior (2ª a 5ª).

La opción 6 permite la carga de cualquier rutina de ordenación anteriormente expuesta, sin interruptir la ejecución del programa, para ello se utiliza la orden CHAIN MERGE, línea 2530.

El listado es el siguiente:

```
60                                                    M E N U
70 PRINT CLS$
80 PRINT TAB(25) M E N U
90 PRINT TAB(24) ==========
100 PRINT
110 PRINT TAB(20) 1  Entrada de datos"
120 PRINT TAB(20) 2  Grabar fichero
130 PRINT TAB(20) 3  Leer fichero"
140 PRINT TAB(20) 4  Ordenacion"
150 PRINT TAB(20) 5  Salida de datos'
155   Salir de rutina de ordenacion
160 PRINT TAB(20)'6  Cargar rutina de ordenacion"
170 PRINT PRINT PRINT·PRINT
180 INPUT  S E L E C C I O N A R   O P C I O N  (1-5) ',OP
190 IF OP=6 THEN PRINT CLS$ PRINT"O R D E N A N D O   L I S T A   GOTO 2500
200 ON OP GOSUB 300,500,700,1000,2000
210 GOTO 60
```

2ª ENTRADA DE DATOS (ENTRADAT). Permite la introducción de datos por teclado —de una lista de la dimensión deseada— tanto numéricos como alfanu-

méricos. Este tipo de entradas fue estudiado en el capítulo 2°, no incorpora depuración de los mismos, por lo que deben ser introducidos con cierto cuidado (puede añadirse una rutina de depuración de las estudiadas en dicho capítulo 2).

```
300 '                                   RUTINA DE ENTRADA DE DATOS
310 '                                            ENTRADAT  ..
320 LON=0·PRINT CLS$
330 INPUT "No  de datos de la lista ";N
340 ERASE A$:DIM A$(N)
350 A$(0)=STR$(N)
360 FOR K=1 TO N
370 PRINT"DATO";K;:INPUT D$
380 IF VAL(D$) THEN IF LEN(D$)>LON THEN LON=LEN(D$)
390 A$(K)=D$
400 NEXT K
410 '   .   ..   ..    .    Homogeneizar datos (igual longitud)
420 FOR k=1 TO N:IF VAL(A$(K))=0 AND A$(K)<>"0" THEN 440
430 A$(K)=STRING$(LON-LEN(A$(K)),"0")+A$(K)
440 NEXT K
450 PRINT"DATOS ALMACENADOS EN MEMORIA"
460 PRINT:INPUT "PRESIONAR <RETURN> PARA CONTINUAR",Z$
470 RETURN
```

Los números han de ser enteros (sin decimales), si se necesita trabajar con números reales, deben multiplicarse previamente o por programa por un valor constante a fin de hacerlos enteros, debido a que el programa trabaja siempre con datos alfanuméricos, a fin de dotarlo de más versatilidad. Por ejemplo si tenemos una lista como la siguiente

0.123 1.21 4.328 0.26 0.04

procedemos a multiplicar los datos por 1000 de forma que quede

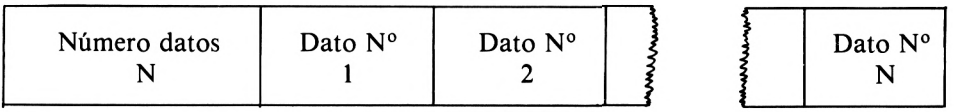
123 1210 4328 260 40

En la salida de resultados puede imprimirse los datos divididos por 1000 y obtener así los verdaderos valores.

3ª GRABAR EN FICHERO (GRAFICHE). Los datos introducidos mediante ENTRADAT son grabados en el disco, de forma secuencial. (Ver ficheros secuenciales).

En primer lugar se guarda el número de datos que contiene la lista (N) y después los datos propiamente dichos.

De esta forma, el valor N también se incluye en el fichero.



El nombre del fichero se establece mediante INPUT.

```
500 '                    .      . . .     .        GRABACION DE DATOS EN FICHERO
510 '                                                      GRAFICHE
520 PRINT cls$:PRINT"GRABACION DE DATOS EN FICHERO"·PRINT
530 INPUT "NOMBRE DEL FICHERO ";F$
540 OPEN "O",1,F$
550 PRINT:PRINT:PRINT"G  R  A  B  A  N  D  O   FICHERO ";F$
560 WRITE #1,A$(Ø)
570 FOR K=1 TO N
580 WRITE #1,A$(K)
590 NEXT K
600 CLOSE
610 PRINT"FICHERO  ";F$;"  GRABADO  PRESIONAR <ENTER> PARA CONTINUAR"; INPUT  ""
Z$
620 RETURN
```

4ª LEER DATOS DE UN FICHERO (LEEFICHE). Permite realizar la operación contraria a la rutina anterior, es decir, recuperar los datos guardados en el disco.

Para poder realizar la lectura se pide el nombre del fichero, que debe ser el mismo que se utilizó al grabar.

Como el primer dato del archivo es el número de elementos, los bucles y dimensionamiento de la matriz (lista) que va a recibir los datos se realiza en función de dicho número, leído del fichero por la variable (N).

```
700 '                    LECTURA DE DATOS DE UN FICHERO
710 '                              LEEFICHE
720 PRINT CLS$
730 INPUT "NOMBRE DEL FICHERO ",F$
740 OPEN "I",1,F$
750 INPUT #1,N$ N=VAL(N$)
760 ERASE A$ DIM A$(N) A$(0)=N$
770 FOR K=1 TO N
780 INPUT #1,A$(K)
790 NEXT K
800 CLOSE
810 PRINT"FICHERO ",F$," LEIDO  PRESIONE <ENTER> PARA CONTINUAR ", INPUT "",Z$
820 RETURN
```

5ª ORDENACION (ORDENAn). Esta rutina puede ser cualquiera de las incluidas en los apartados precedentes.

Los nombres dados a dichas rutinas es ORDENA, seguido de un número que las identifica y cuya relación incluimos a continuación.

Nombre	*Método*
ORDENA1	Intercambio
ORDENA2	BUBBLE
ORDENA3	BUBBLE con marca
ORDENA4	Intercambio retardado
ORDENA5	Doble pasada
ORDENA6	SHELL-METZNER (1)
ORDENA7	SHELL-METZNER (2)
ORDENA8	QUICK-SORT

6ª SALIDA DE DATOS ORDENADOS (SALDATOS). Constituye una sencilla rutina de salida de datos ordenados. En caso de que el número de datos sea mayor de 20, se interrupte el scrolling automático de pantalla, a fin de observarlos con detenimiento, y presionando ENTER se produce la salida del siguiente bloque de 20 datos. (línea 2070).

Las líneas 2020 y 2080 permiten que la salida de datos se realice también por impresora.

```
2000                                    RUTINA SALIDA DE DATOS
2010                                          SALDAT
2020 PRINT CLS$ INPUT "SALIDA POR IMPRESORA (S/N) ",I$
2030 PRINT CLS$
2040 PRINT"DATOS ORDENADOS"
2050 PRINT"================"
2060 FOR K=1 TO N
2070 IF K/20=INT(K/20) AND UPPER$(I$)<>"S" THEN INPUT "PRESIONAR <RETURN> PARA
ONTINUAR",Z$
2080 IF UPPER$ (I$)="S" THEN LPRINT A$(K)
2090 PRINT A$(K)
2100 NEXT K
2110 PRINT INPUT "PRESIONAR <RETURN> PARA CONTINUAR",Z$
2120 RETURN
```

Conjunto Ensamblado

Por último y para mayor comodidad del lector, se incluye un programa con todas las rutinas mencionadas, convenientemente ensambladas y regidas desde el menú.

Como observará, se han añadido algunas pequeñas modificaciones con el fin de que puedan funcionar en conjunto.

Para probar cómodamente todos los métodos de ordenación descritos, e incluso cronometrar el tiempo invertido por cada uno, proceda de la siguiente forma.

1. Grabe las rutinas de ordenación con el nombre que se incluye en la segunda línea de cada listado.
2. Grabe el programa que incluimos al final con las rutinas de trabajo ensambladas, tal y como está en el listado. El nombre indicado para este programa es ORDEN.
3. Haga RUN"ORDEN o RUN simplemente si lo tiene ya en memoria, para ejecutar el programa.

   Una vez aparezca el menú, seleccione la opción 6, introduciendo —6<ENTER>—. El programa le pedirá el nombre de la rutina de ordenación que desea utilizar; introduzca el nombre según figura en el epígrafe 5° (si los grabó con esos mismos).

   Una vez de vuelta al menú, introduzca una lista de datos numéricos (del tamaño que desee) totalmente desordenados. (También puede usar un método aleatorio sustituyendo el INPUT de ENTRADAT por un RND).
4. Cuando retorne al MENU, seleccione la opción 2 para grabarlos en el disco. Como nombre puede utilizar "NUMEROS" o cualquier otro.

5. Seleccione nuevamente la opción 1 e introduzca ahora una lista de datos alfanuméricos. Después la 2 para grabarlos. Un nombre apropiado puede ser, por ejemplo, "PALABRAS".
6. Ahora puede proceder a ordenar cualquiera de los dos ficheros creados. Si es precisamente el último, éste se encontrará ya en memoria; por tanto, basta con que seleccione la opción 4 y después la 5 con el fin de verlos ordenados.
7. Si desea ordenar un archivo que está contenido en disco utilice la opción 3 para leerlos, luego 4 para ordenar y 5 para imprimirlos en pantalla, ya ordenados.

Tenga en cuenta que todas las rutinas retornan al MENU al finalizar su tarea.

Notas:

1 En todos los métodos de ordenación expuestos, se ha considerado que ésta se realiza de menor a mayor. Para obtener clasificaciones en sentido contrario basta con cambiar los operadores de relación de los IF, o bien, simplemente cambie el bucle de salida de datos en la forma.

FOR K = N TO 1 STEP—1
(Línea 2060 de la rutina SALDATOS)

2 Si desea salvar los datos ordenados tras las operaciones pertinentes, utilice la opción 2 (grabar), ya que tras la opción 4 (ordenar), en memoria estarán todos los datos ordenados.

```
10 ´ CONJUNTO DE RUTINAS PARA ENSAMBLAR CON PGM  DE ORDENACION
20 ´  .                          ORDEN
30 DIM A$(10),X(2,1)
40 CLS$=CHR$(27)+"E"+CHR$(27)+"H"
50 PRINT CLS$
60 ´                                                M E N U
70 PRINT CLS$
80 PRINT TAB(25)"M E N U"
90 PRINT TAB(24)"==========="
100 PRINT
110 PRINT TAB(20)"1 Entrada de datos"
120 PRINT TAB(20)"2  Grabar fichero"
130 PRINT TAB(20)"3. Leer fichero"
140 PRINT TAB(20)"4  Ordenacion"
150 PRINT TAB(20)"5. Salida de datos"
155 ´ Salir de rutina de ordenacion
160 PRINT TAB(20)"6  Cargar rutina de ordenacion"
170 PRINT PRINT PRINT PRINT
180 INPUT "S E L E C C I O N A R   O P C I O N  (1-5) ",OP
190 IF OP=6 THEN PRINT CLS$ PRINT"O R D E N A N D O   L I S T A"  GOTO 2500
200 ON OP GOSUB 300,500,700,1000,2000
210 GOTO 60
300 ´     ..                          RUTINA DE ENTRADA DE DATOS
310 ´.. .  ..                                  ENTRADAT
320 LON=0:PRINT CLS$
330 INPUT "No  de datos de la lista ",N
340 ERASE A$:DIM A$(N)
350 A$(0)=STR$(N)
360 FOR K=1 TO N
370 PRINT"DATO",K, INPUT D$
380 IF VAL(D$) THEN IF LEN(D$)>LON THEN LON=LEN(D$)
390 A$(K)=D$
400 NEXT K
```

```
410 '                          Homogeneizar datos (igual longitud)
420 FOR k=1 TO N IF VAL(A$(K))=0 AND A$(K)<>"0" THEN 440
430 A$(K)=STRING$(LON-LEN(A$(K)),"0")+A$(K)
440 NEXT K
450 PRINT"DATOS ALMACENADOS EN MEMORIA"
460 PRINT INPUT "PRESIONAR <RETURN> PARA CONTINUAR",Z$
470 RETURN
500 '                                    GRABACION DE DATOS EN FICHERO
510 '                                           GRAFICHE
520 PRINT cls$ PRINT"GRABACION DE DATOS EN FICHERO" PRINT
530 INPUT "NOMBRE DEL FICHERO ",F$
540 OPEN "O",1,F$
550 PRINT PRINT PRINT"G  R  A  B  A  N  D  O   FICHERO ",F$
560 WRITE #1,A$(0)
570 FOR K=1 TO N
580 WRITE #1,A$(K)
590 NEXT K
600 CLOSE
610 PRINT"FICHERO  ",F$,"  GRABADO  PRESIONAR <ENTER> PARA CONTINUAR", INPUT ""
Z$
620 RETURN
700 '                 LECTURA DE DATOS DE UN FICHERO
710 '                          LEEFICHE
720 PRINT CLS$
730 INPUT "NOMBRE DEL FICHERO ",F$
740 OPEN "I",1,F$
750 INPUT #1,N$ N=VAL(N$)
760 ERASE A$ DIM A$(N) A$(0)=N$
770 FOR K=1 TO N
780 INPUT #1,A$(K)
790 NEXT K
800 CLOSE
810 PRINT"FICHERO ",F$," LEIDO  PRESIONE <ENTER> PARA CONTINUAR ", INPUT "",Z$
820 RETURN
1000 ' RUTINA DE ORDENACION A ENSAMBLAR
1020 '
1030 '
1040 '
1050 '
1500 ' Salida de la rutina de ordenacion
1510 PRINT INPUT"DATOS ORDENADOS  PRESIONE <ENTER> PARA CONTINUAR ",Z$
1520 GOTO 70
2000 '                                RUTINA SALIDA DE DATOS
2010 '                                       SALDAT
2020 PRINT CLS$ INPUT "SALIDA POR IMPRESORA (S/N) ",I$
2030 PRINT CLS$
2040 PRINT"DATOS ORDENADOS"
2050 PRINT"==============="
2060 FOR K=1 TO N
2070 IF K/20=INT(K/20) AND UPPER$(I$)<>"S" THEN INPUT "PRESIONAR <RETURN> PARA
ONTINUAR",Z$
2080 IF UPPER$ (I$)="S" THEN LPRINT A$(K)
2090 PRINT A$(K)
2100 NEXT K
2110 PRINT INPUT "PRESIONAR <RETURN> PARA CONTINUAR",Z$
2120 RETURN
2500 '              CARGAR RUTINA DE ORDENACION
2510 PRINT CLS$
2520 INPUT "NOMBRE DE LA RUTINA DE ORDENACION ",P$
2530 CHAIN MERGE P$,100,ALL
```

10. PROGRAMA ORDENACION DE FECHAS

En programas de gestión es frecuente la creación de archivos, en los cuales se va depositando información, intimamente ligada a la fecha en la cual se produjo dicha información. Dado que dichas fechas no tienen porque ser correlativas respecto a su introducción, cuando se han de manejar datos del fichero, este se encuentra desordenado respecto a sus fechas. Con los métodos expuestos anteriormente pueden ordenarse números y textos (alfabéticamente), pero el caso que nos ocupa es de naturaleza distinta.

Si consideramos un formato de fecha muy extendido como es la introducción del día, mes y año con dos caracteres, en la forma

DD/MM/AA

puede observarse que un conjunto de fechas como

1ª 15/10/87
2ª 12/09/83
3ª 01/12/85
4ª 02/02/87

y aplicando cualquiera de los métodos expuestos, se obtendría el conjunto ordenado (de menor a mayor)

1ª 01/12/85
2ª 02/02/87 mal ordenados
3ª 12/09/83
4ª 15/10/87 bien ordenados

Las fechas 1ª y 3ª están mal ordenadas puesto que eldía de la 1ª es "01" y de la 2ª "12", el ordenador no sigue considerando el resto de los caracteres. En conclusión el método no es válido, es necesario buscar "algo", que identifique de forma inequívoca a cada fecha y que sea comparable por el ordenador a efectos de ordenación.

Existen varios algoritmos matemáticos que permiten transformar fechas en números que las identifican, entre ellos el calculo del "código de fecha", mediante la expresión.

$$C = INT\ (365.25*A2) + INT\ (30.56*M) + D + N$$

donde

A2 = dos últimos dígitos del año (p.e. 1987; A2 = 87)
M = número que identifica al mes (1 a 12)
D = número del día (1 a 31)

$$N = \begin{cases} N=0 & \text{(valor más general)} \\ \text{Si } M=1 \text{ ó } M=2 \text{ (enero o feb.)} & \begin{cases} N=2 \text{ si el año es bisiesto} \\ N=1 \text{ si el año no es bisiesto} \end{cases} \end{cases}$$

De una fecha dada, con el formato descrito, los valores de A2, M y D son inmediatos, por ejemplo

15/09/75

D = 15 M = 9 A2 = 75

Si consideramos que dicha fecha estará almacenada como string (cadena), puede leerse mediante una variable alfanumérica p.e. F$, es decir, F$ = "15/09/75".

La obtención de D, M y A2 se realiza fragmentando la cadena, y aplicando a las tres subcadenas resultantes la función VAL (peso en valor numérico).

```
A2 = VAL (RIGHT$ (F$,2))
M  = VAL (MID$ (F$,4,2))
D  = VAL (LEFT$ (F$,2))
```

solo queda pues calcular el valor de N que depende del mes y de que el año sea bisiesto o no.

Verificar si M = 1 ó M = 2, no presenta dificultades basta con un

IF M = 1 OR M = 2 —si→ N = 1 ó 2 (depende de que el año sea o no bisiesto)
↓ no
N = 0

La condición de bisiesto, es que A2 sea distinto de 0 y que además A2 sea divisible por 4.

Si A2 es divisible por 4, se cumplirá

A2/4 = INT (A2/4)

Por tanto la condición de bisiesto puede escribirse en BASIC

IF A2 < > 0 AND A2/4 = INT (A214) —si→ bisiesto
↓ no
no es bisiesto

El valor de C así obtenido es único para cada fecha y menor cuanto menor es la fecha en el tiempo (más antigua), por tanto puede pues realizarse una ordenación sobre el conjunto de códigos y simultáneamente ordenar las fechas.

El siguiente programa, utiliza las expresiones utilizadas anteriormente y permite la ordenación de un número determinado de fechas (dado por INPUT). El método de ordenación utilizado es el más sencillo, intercambio, aunque puede utilizarse cualquier otro cambiando las líneas 200 a 280.

```
10 ´ ORDENACION DE FECHAS (INTERCAMBIO)
20 ´ Entrada de datos
25 PRINT"Formato de fechas  dd/mm/aa"
26 PRINT "==========================="
30 INPUT "Numero de datos ",num DIM F$(num),C(num)
40 FOR k=1 TO num
50 PRINT"fecha";k,
60 INPUT f$(k)
70 NEXT k
100 ´ Calculo y almacenamiento de codigos
110 FOR k=1 TO num
120 F$=F$(K)
130 GOSUB 500
140 C(K)=C
150 NEXT K
200 ´ Ordenacion de codigos y fechas (por intercambio)
210 FOR I=1 TO num-1
220 FOR J=1 TO num-1
230 IF C(J)<=C(J+1) THEN 270
240 C=C(J) F$=F$(J)
250 C(J)=C(J+1) F$(J)=F$(J+1)
260 C(J+1)=C F$(J+1)=F$
270 NEXT J
280 NEXT I
300 ´ Salida de fechas ordenadas
310 PRINT" FECHA          Codigo"
315 PRINT "----------------------"
320 FOR K=1 TO num
330 PRINT F$(K),C(K)
340 NEXT K
350 END
500 ´ Calculo del codigo de dia
510 d=VAL(LEFT$(F$,2))
520 m=VAL(MID$(F$,4,2))
530 a=VAL(RIGHT$(F$,2))
540 IF m=1 OR m=2 THEN GOSUB 570 ELSE n=0
550 C=INT(365 25*a)+INT(30 56*m)+d+n
560 RETURN
570 IF a<>0 AND a/4=INT(a/4) THEN n=1 ELSE n=2
580 RETURN
```

Obsérvese como en los intercambios (líneas 240 a 260), a la vez que se ordenan los códigos (según su valor), se ordenan las fechas. Código y fecha del mismo índice van intimamente unidos.

Si como es normal las fechas llevasen emparejados uno o varios datos, estos se intercambiarían también dentro de dichas líneas. Por ejemplo, cada fecha va unida a una cantidad y a un identificador de cargo o abono (D/H). Podía considerarse por el programa que existen 4 matrices (de una dimensión).

C ( ) códigos	CAN ( )cantidades
F$ ( ) fechas	IDS ( ) identificador cargo/abono

Las líneas de transformación serían

```
240 C = C(J) : F$ = F$(J) : CAN = CAN(J) ; ID$ = ID$(J)
250 C(J) = C(J + 1) : F$(J) = F$(J + 1) : CAN(J) = CAN(J + 1) : IDS(J) = IDS(J + 1)
260 ((J + 1) = C : F$(J + 1) = F$ : CAN(J + 1) = CAN : ID$(J + 1) = ID$
```

siendo C, F$, CAN e ID, variables auxiliares para que el intercambio se realice dentro de las propias matrices.

En la salida de resultados (300-340) se ha indicado además de la fecha su código, únicamente a título de ejemplo y comprobación. En el caso expuesto anteriormente se puede modificar la línea 330 en la forma

```
330 PRINT F$(K), CAN(K), ID$(K), C(K)
```

Lógicamente también habrá de modificarse convenientemente la cabecera, líneas 310 y 315.

CAPITULO 4

INTRODUCCION A LOS FICHEROS

1. DEFINICION DE FICHERO

2. TIPOS DE FICHERO
* Secuenciales
* Indexados
* Acceso directo

3. CONCEPTO DE REGISTRO Y CAMPO

4. Introducción a los ficheros

1. DEFINICION DE FICHERO

Siempre resulta arriesgado dar una definición, por ello me limitaré a indicar que un fichero es un conjunto de datos relacionados entre sí, generalmente almacenado en un soporte magnético, disco o cinta. También cabe definirlo como conjunto de fichas o registros de datos por analogía con los ficheros o clasificadores de todos conocidos.

A cada fichero o conjunto de información se le asigna un nombre por el que será reconocido, de forma que toda transmisión de datos entre la memoria del ordenador y el fichero, o viceversa, exige la referencia a dicho nombre.

2. TIPOS DE FICHERO

Pueden realizarse diversas clasificaciones de ficheros, según su contenido, tipo de acceso, tiempo de existencia, etc.

Respecto a su contenido, los ficheros pueden clasificarse como:

Numéricos	(datos numéricos)
Alfanuméricos	(datos alfanuméricos, cadenas)
Mixtos	(datos numéricos y alfanuméricos)

Por otro lado, los datos, sea cual sea su naturaleza, pueden guardarse de distinta forma, lo cual da origen a una nueva clasificación:

Forma de almacenamiento de los datos	Ficheros binarios Ficheros hexadecimales Ficheros octales Ficheros ASCII

Los ficheros binarios son aquellos cuya información es guardada en números binarios, es decir, en base 2; por tanto, dicha información se compone únicamente de ceros y unos, que son los dígitos disponibles en este sistema de numeración.

De igual forma, en los ficheros hexadecimales y octales la información está guardada en números de base 16 y 8, respectivamente.

Los ficheros ASCII son aquéllos cuya información es guardada de acuerdo con un código estándar cuyas siglas corresponden al "American Standard Code for Information Interchange" (Código Americano Normalizado para el Intercambio de In-

formación). Esta norma permite la codificación de caracteres —un total de 128 signos— que puede ver en el manual de su ordenador.

De acuerdo con la forma en que se accede a los datos, los ficheros pueden ser:

* Secuenciales
* Indexados
* Aleatorios o de acceso directo

Ficheros secuenciales

Este tipo de ficheros va ligado íntimamente a la forma en que una cinta cassete guarda la información. Los datos son almacenados uno tras otro "secuencialmente" a medida que son introducidos.

El principal inconveniente de este método de almacenamiento, reside en que la lectura debe realizarse también secuencialmente; es decir, para leer un dato determinado, hay que leer antes todos los anteriores (como mínimo).

No todo son inconvenientes, su sencillez es un punto a su favor, así como la posibilidad de almacenar datos de muy diversa longitud, coexistiendo perfectamente en el fichero, cosa imposible en los de acceso directo.

Ficheros indexados (acceso por claves)

Permiten el acceso directo a un dato determinado, a través de un parámetro —índice— que permite la búsqueda al ordenador a través de una tabla de índices. Su gran ventaja es que la búsqueda puede ser referida a un determinado campo dentro de un registro (ver estos conceptos).

Ficheros de acceso directo

El acceso a un dato determinado es directo, entendiendo como dato un registro, no un campo. El ordenador, mediante un algoritmo, calcula la posición que ocupa en el disco el dato indicado, que lleva un parámetro para indicar el número de orden dentro del conjunto de datos.

Atendiendo al tiempo de existencia de los ficheros, pueden clasificarse, no exhaustivamente, entre tres grandes grupos:

Transitorios (corto plazo)
Temporales (medio plazo)
Fijos (largo plazo)

Los ficheros transitorios son aquellos cuya vida es muy corta. Generalmente duran sólo el tiempo en que están sujetos a modificación o creación, consolidándose posteriormente como temporales.

Los temporales, que son la inmensa mayoría, son ficheros dinámicos sujetos a continuas variaciones (actualización). Por último los fijos, son aquellos que aún siendo susceptibles de modificaciones, éstas no son frecuentes, por lo que en un plazo medio pueden considerarse inalterables.

3. CONCEPTO DE REGISTRO Y CAMPO

Observando la figura que simula una ficha convencional de un fichero de alumnos, se desprende que para cada alumno existen una serie de epígrafes ubicados de una forma fija, que una vez completados constituyen los datos del mismo.

COLEGIO XXX

Alumno _____

Domicilio _____

Edad _____ Tel. _____

Curso _____

Pues bien, si por analogía, entendemos que un fichero físico es igual a uno informático, donde únicamente varía la forma de almacenar la información, el concepto de "ficha" se correspondería con el de "registro" y cada uno de los "epígrafes" de la ficha sería un "campo".

Fichero físico	*Fichero informático*	*Tipo de información*
Ficha	Registro	Varios datos
Epígrafe	Campo	Un solo dato

Es decir, volviendo al ejemplo, la ficha del alumno XXX sería un registro del fichero, mientras que el nombre del alumno será el primer campo de ese registro.

La información de una ficha quedaría almacenada en un fichero en la forma que indica la figura, puesto que ésta es depositada linealmente en el soporte de archivo (disco).

Campo 1	Campo 2	Campo 3	Campo 4	Campo 5
Nombre	Domicilio	Edad	Tel.	Curso

Registro N° n

Un registro puede contener uno o más campos, al igual que un fichero puede contener uno o más registros, lo habitual es obviamente, ficheros conteniendo varios registros y a su vez, varios campos.

En los capítulos siguientes se verán en profundidad los conceptos expuestos hasta ahora.

CAPITULO 5

FICHEROS SECUENCIALES

1. INTRODUCCION

2. APERTURA Y CIERRE DE FICHEROS

3. LECTURA Y ESCRITURA

4. FINAL DE FICHERO

5. TIPOS DE LECTURA

6. TIPOS DE ESCRITURA

7. EXISTENCIA DE UN FICHERO

8. EJEMPLO PRACTICO 1
   Gestión de bibliotecas

9. EJEMPLO PRACTICO 2
   Juego educativo de preguntas y respuestas

5. Ficheros secuenciales

1. INTRODUCCION

Como se indicó en el capítulo anterior, en este tipo de ficheros la información es almacenada de forma "secuencial", es decir, los datos se guardan uno a continuación de otro, en la misma forma, en que se encuentran en una lista. Su principal inconveniente consiste en que el acceso a un dato no es directo, sino que deben leerse previamente todos los anteriores hasta llegar el indicado. Es como si en una casete de música, se deseará buscar una determinada palabra, para ello debe escucharse dicha cinta desde el principio.

A pesar de que este inconveniente los invalida para ciertos trabajos, su sencillez de creación y manejo y su cierto grado de analogía con las matrices, les hace especialmente indicados en muchos casos.

2. APERTURA Y CIERRE DE FICHEROS

Existen dos conceptos fundamentales en el manejo de ficheros, aplicables también a los secuenciales

1° Abrir el fichero
2° Cerrar el fichero

Antes de manipular un fichero, es necesario proceder a su apertura, esta apertura puede realizarse con dos objetivos: lectura o escritura, es decir, puede abrirse un fichero con el fin de leer datos contenidos en él, o bien, abrirlo para introducir datos.

Aparentemente esta diferencia puede parecer un tanto artificiosa, bastaría simplemente con abrirlo, sin especificar que operación se va a realizar con él, pero hay que tener en cuenta que la apertura para escribir lleva implícita la creación del propio fichero si este no existe en el disco, mientras que la apertura para leer se realiza forzosamente sobre un fichero existente.

Abrir un fichero	Para escribir	{ Creación del fichero si no existe
	Para leer	{ El fichero debe existir

En cuanto al cierre no existe la dualidad anterior, puesto que para cerrar un fichero es necesario que este exista. Unicamente cabe distinguir el caso de la existencia de varios ficheros abiertos simultáneamente, lo cual daría origen a una discriminación en el cierre, es decir, puede cerrarse uno u otro en particular o todos en general.

Cerrar un fichero	Cerrar uno determinado
	Cerrar todos

Vamos a ver como se articulan desde el Mallard BASIC todos estos conceptos.

*** Abrir un fichero**

Un fichero es una estructura de datos, cerrada, dotada de una única puerta de acceso y con un final físico o fondo de saco.

La apertura de la puerta de acceso a un fichero se realiza mediante una clave, dicha clave se denomina "Nombre del fichero", por lo que cualquier referencia al mismo, debe hacerse por dicho nombre.

La forma general de apertura de ficheros secuenciales es:

OPEN modo, nº de fichero, nombre

donde

— *Modo.* Es el literal "O" ó "I" que especifica el destino de la apertura, para escribir o leer respectivamente.

"O" (output) Abrir para escribir
" I " (input) Abrir para leer

— *Nº de fichero.* Puesto que pueden existir varios ficheros abiertos simultáneamente es necesario identificar cada uno con un número, por el cual será reconocido en lo sucesivo.

— *Nombre.* Es la palabra clave que identifica al fichero en el disco. El nombre se puede dar en dos formas

nombre	literal _____	p.e. "DATOS1"
	variable alfanumérica _____	p.e. F$ donde F$ contiene el nombre del fichero

La máxima longitud de un nombre es 8 caracteres más 3 opcionales separados por un punto, al igual que los nombres de programas.

Observe que cada fichero lleva dos identificadores: Nombre y número; el nombre lo identifica dentro del disco (directorio) y el número lo hace dentro del programa.

Ejemplos: En un programa se necesitan realizar las siguientes operaciones de apertura.

1) Abrir fichero SALARIOS para escribir
2) Abrir fichero HORAS para escribir
3) Abrir fichero NOMBRES para leer

Las órdenes serían sucesivamente.

```
OPEN "O", 1, "SALARIOS"      (ver nota 3)
OPEN "O", 2, "HORAS"
OPEN "I", 3, "NOMBRES"
```

Notas:

1ª Si se abre un fichero para leer, es necesario que este exista en el disco en curso, en caso contrario se producirá un error.
2ª Si se abre un fichero para escribir y este no existe, es creado de forma automática.
3ª Si se abre un fichero para escribir y este existe, es borrado automáticamente para crear uno nuevo con el mismo nombre.
4ª En principio sólo pueden abrirse 3 ficheros simultáneamente.

*** Cerrar un fichero**

El cierre de un fichero presupone, al menos de momento, que no se van a realizar más operaciones de lectura/escritura sobre él.

La forma general es

```
CLOSE Lista de : nº de fichero
```

La simplicidad de la orden es obvia, si cada fichero se identifica en la apertura con un número, basta con referenciar dicho número. El cierre es independiente del tipo de operación I/O.

En la ejecución de esta orden deben tenerse en cuenta las siguientes cuestiones.

1ª) Si se omite la lista de números de ficheros, se produce el cierre de todos los que estuvieran abiertos

```
CLOSE    Cierra todos los ficheros
```

2ª) Si se cierra un fichero que se abrió para escritura ("O"), se grabarán todos los datos contenidos en el tampón de salida y posteriormente el cierre propiamente dicho. Tenga en cuenta que los datos no se graban uno a uno, a medida que se producen las órdenes de escritura, sino por bloques, con el fin de minimizar el número de accesos al disco.

3ª) Si el número de fichero indicado en CLOSE no estaba asignado previamente por una orden de lectura, dicho número es ignorado

3. LECTURA Y ESCRITURA

Dos son las operaciones que pueden realizarse sobre un fichero, la lectura de datos existentes en el mismo o la escritura de datos en él, ambas operaciones se reflejan en órdenes distintas.

*** Lectura de datos de un fichero**

La operación de lectura se realiza con la palabra INPUT, al igual que la lectura desde el teclado, seguida del indicador #, propio de disco.

Su forma general es:

INPUT # nº de fichero, lista de : variable

NOTA La nomenclatura Lista de variable indica que en ese lugar puede incluirse una variable o varias separadas por comas Ejemplo INPUT # 1, A, Z, X (1 es el numero del fichero y A, Z y X la lista de variables)

El número de fichero es que le identifica y corresponde al dado en la orden OPEN. La lista de variables son aquellas variables en las que se almacenarán los datos leidos, dicha lista puede estar constituida por una sola variable o varias y el tipo de variable corresponderá al tipo de dato a leer (número entero, real o cadena).

En el caso de tener que leer una línea completa de un fichero debe usarse LINE INPUT #, en vez de INPUT #. Tenga en cuenta que LINE INPUT tanto en teclado como en fichero admite leer comas dentro de las cadenas, sin ser interpretadas como separadores de datos, sino como parte del texto.

Como veremos más adelante las variables más utilizadas para leer gran cantidad de datos de un fichero, son las indexadas (matrices) por su capacidad de almacenamiento y fácil manipulación mediante bucles FOR ó WHILE así como por su analogía con los propios ficheros secuenciales.

*** Escritura de datos en un fichero**

La operación de escritura viene regida por la orden WRITE # y su forma general es la misma que la de lectura, a fin de compatibilizar ambas operaciones.

WRITE # nº de fichero, lista de : variable ó expresión

Si se utiliza la opción "variable" se guarda el contenido de la variable. Si se utiliza "expresión" se guardará el resultado de dicha expresión. Por ejemplo si trabajamos con cadenas el mismo efecto producirían las 2 líneas siguientes

```
A$ = "HOLA" : WRITE #1, A$          (variable)
WRITE #1, "HOLA"                    (expresión)
```

También puede utilizarse la orden PRINT# con el mismo formato, pero existen algunas diferencias: la orden WRITE# ignora las "zonas" de escritura (pantalla), escribe las cadenas entre comillas, separa los elementos por comas, el separador final de la orden no produce efecto en la grabación. En principio use solo esta última.

4. FINAL DE FICHERO

La función EOF permite conocer en cada momento el proceso de lectura de un fichero, si no se ha alcanzado el final devuelve el valor cero (0), mientras que si se ha llegado al agotamiento de los datos, genera el valor menos uno (—1).

Valor de EOF	*Significado*	*Comentario*
—1	verdadero	Se ha alcanzado el final del fichero
0	falso	No se ha alcanzado su final

Esta señal como veremos a continuación permite la lectura completa de un fichero del que se desconoce el número de elementos que lo componen.

5. TIPOS DE LECTURA

Son varias las estructuras o rutinas de lectura de datos de un fichero. Dependiendo del conocimiento sobre el número de datos que contiene; podemos encontrarnos en tres casos.

Tipos de lectura
- Nº datos conocidos
- Nº datos contenido en el propio fichero
- Nº datos desconocido

Cada uno de estos casos puede resolverse por aplicación de distintas rutinas basadas en distintos tipos de bucles, FOR, WHILE o contador + IF.

A continuación se exponen varias rutinas de lectura que abarcan algunas de las muchas posibles soluciones.

*** Número datos conocido. Bucle FOR**

Esta rutina está preparada para leer el fichero "DATOS", que contiene 30 elementos o más, todos ellos numéricos (enteros o reales). Si el fichero contiene menos de

30 elementos se producirá un error de lectura tipificado como EOF met in 60 indicando que se ha alcanzado el final del fichero.

```
10 ' RUTINA DE LECTURA FOR, NUM, DATOS CONOCIDO (LET)
15 ' ................... FS1 ..........................
20 N=30:DIM X(N)
30 PRINT "Lectura de 30 Datos del fichero <DATOS>"
40  OPEN "I",1,"datos"
50  FOR K=1 TO N
60 INPUT #1,X(K):PRINT X(K)
70  NEXT K
80  CLOSE
90 PRINT "Lectura terminada"
```

El bucle FOR de la línea 40 se extiende al valor 1 y al valor N (30), que es el número de datos del fichero. La rutina presupone el conocimiento previo de N y del nombre del fichero, por tanto posee un alto grado de rigidez y especialización. A fin de generalizar el caso, podía modificarse en la forma siguiente.

```
10 ' RUTINA DE LECTURA FOR, NUM, DATOS CONOCIDO (INPUT)
15 ' ................... FS2 ..........................
20 INPUT "Nombre del fichero ";F$
30 INPUT "Numero de datos del fichero";N
40 DIM X(N)
50  OPEN "I",1,F$
60  FOR K=1 TO N
70 INPUT #1,X(K):PRINT X(K)
80  NEXT K
90  CLOSE
100 PRINT "Lectura terminada"
```

De esta forma queda variable el fichero y el número de datos a leer.

Obviamente la utilización de cualquier rutina de lectura está condicionada a la existencia del fichero de datos en el disco (ver epígrafe «Tipos de escritura»).

Nota: El símbolo Pt que aparece en los listados corresponde a #.

*** Número de datos conocidos. Bucle contador**

Aunque la estructura FOR es cómoda, puede sustituirse también por un contador controlado por un IF, es decir, los programas que exponemos a continuación son equivalentes.

```
10 FOR K=1 TO 100
20 PRINT K
30 NEXT K
40 END
```

```
10 K=1
20 PRINT K
30 K=K+1: ´Contador
40 IF K>100 THEN END
50 GOTO 20
```

Ambos programas, como puede comprobar, escriben los números del 1 al 100.

Basados en esta idea, puede construirse una rutina similar a la anterior (FS2), cambiando las líneas 60 en adelante en la forma.

```
10 ' RUTINA LECTURA BUCLE CONTADOR, NUM, DATOS CONOCIDO (INPUT)
15 ' ................. FS3 ........................
20 INPUT "Nombre del fichero ";F$
30 INPUT "Numero de datos del fichero";N
40 DIM X(N)
50 CON=1
60  OPEN "I",1,F$
70 IF CON > N THEN 110
80 INPUT #1,X(CON):PRINT "Dato No.";CON;X(CON)
90 CON=CON+1
100 GOTO 70
110 CLOSE
120 PRINT "Lectura terminada"
```

*** Número de datos contenido en el propio fichero**

Es costumbre muy útil el grabar en un fichero secuencial, el número de datos que contiene en primer lugar, de forma que esta información puede ser recogida por el proceso de lectura, automatizando dicho proceso.

El método a utilizar puede ser cualquiera de los dos descritos, contador o bucle FOR, e incluso un bucle WHILE, aunque reservamos este último para ficheros con el número de datos desconocido, por ser el más idóneo.

Una rutina construida con FOR podía ser:

```
10 ' RUTINA LECTURA BUCLE FOR, NUM, DATOS EN FICHERO
15 ' ................. FS4 ........................
20 INPUT "Nombre del fichero ";F$
30  OPEN "I",1,F$
40 INPUT #1,N:DIM X(N)
50 FOR k=1 TO N
60 INPUT #1,X(K):PRINT "Dato No.";K;X(K)
70 NEXT K
80 CLOSE
90 PRINT "Lectura terminada"
```

Dejamos en manos del lector la aplicación de un contador controlado por IF, como variante a esta rutina.

*** Número de datos desconocido. Bucle WHILE-WEND**

En el apartado 4, de este capítulo, nos hemos referido a la función EOF, pues bien, aprovechando las características de los bucles WHILE; cuyo número de vueltas no es fijo como ocurre con FOR, sino que permanece el bucle mientras se de una determinada condición, se puede elaborar una rutina de lectura, apoyada además en la señal EOF de fin de fichero. Veamos un ejemplo al efecto.

```
10 ' RUTINA LECTURA BUCLE WHILE, NUM, DATOS DESCONOCIDO
15 ' ..................... FS5 ........................
20 INPUT "Nombre del fichero ";F$
30 DIM X(100):K=1
40 OPEN "I",1,F$
50 WHILE NOT EOF(1)
60 INPUT #1,X(K):PRINT"Dato No.";k;X(K)
70 K=K+1
80 WEND
90 CLOSE
100 PRINT "Lectura terminada"
```

La indeterminación del número de datos obliga a sobredimensionar la matriz que recibe los datos del fichero, en el ejemplo se supone que el número de datos no es mayor de 100.

Observe como se utiliza dentro del bucle WHILE el operador lógico NOT, junto a la señal EOF de final de fichero que lleva entre paréntesis el número de identificación del fichero, que corresponde al dado por la orden OPEN.

El conjunto de líneas 50 a 80 podría traducirse por: «Mientras no encuentres la señal de fin de fichero, lee datos, almacénalos en la matriz X( ), e imprímelos en pantalla».

Desde luego podía utilizarse EOF con otra estructura de bucle, como se hizo anteriormente, por ejemplo utilizando FOR sería

```
10 ' RUTINA LECTURA BUCLE FOR, NUM, DATOS DESCONOCIDO
15 ' ..................... FS6 ........................
20 INPUT "Nombre del fichero ";F$
30 DIM X(100)
40 OPEN "I",1,F$
50 FOR k=1 TO 100
60 INPUT #1,X(K):PRINT"Dato No.";k;X(K)
70 IF EOF(1) THEN 90
80 NEXT k
90 CLOSE
100 PRINT "Lectura terminada"
```

*** Lectura de un solo dato**

Hasta el momento, en todos los casos expuestos, la operación de lectura se ha extendido a todo el fichero, cargándose los datos contenidos en él, en una matriz. Si se necesita leer un dato concreto y almacenar todo el fichero, la condición impuesta por la propia naturaleza de los ficheros secuenciales, es que han de leerse todos los datos, entre el primero y el buscado.

Veamos una rutina preparada para leer un determinado dato, bajo la premisa de conocer el número de orden que tiene en el fichero.

```
10 ' RUTINA LECTURA DE UN DATO
15 ' ......... FS7 ............
20 INPUT "Nombre del fichero ";F$
30 INPUT "Numero de orden del dato a leer";N
40 OPEN "I",1,F$
50 FOR k=1 TO N
60 INPUT #1,A$
70 NEXT k
80 CLOSE
90 PRINT "El dato No.";N;" del fichero ";F$;" es ";A$
```

Si no se conociese el número de orden, sino algún atributo, podía establecerse una búsqueda por igualdad o inclusión, en la misma forma en que se realizaba en las matrices. Si no existe la certeza de que el dato buscado exista, conviene utilizar un control de fin de fichero, bien del tipo WHILE NOT EOF, o simplemente con un IF EOF, tal como se utiliza en las rutinas FS5 y FS6 respectivamente.

6. TIPOS DE ESCRITURA (grabación)

Dado que esta operación es la contraria de la lectura, es correcto suponer que los distintos tipos vistos anteriormente, son también aplicables, con las variaciones oportunas.

Si partimos del supuesto de que el número de datos a grabar es conocido puede prepararse una rutina, cuya primera operación de escritura sea precisamente de guardar este número. Suponiendo que los datos a guardar estén contenidos en la matriz A, que el número de datos es N y la dimensión de A es también N, tenemos

```
10 ' RUTINA DE GRABACION
15 ' ........ FS8 .......
20  INPUT "NOMBRE DEL FICHERO";F$
30  OPEN "O",1,F$
40  WRITE #1,N
```

```
50  FOR K=1 TO N
60  WRITE #1,A(K)
70  NEXT K
80  CLOSE
90 PRINT"Grabacion terminada"
```

Esta rutina realiza un volcado total de la matriz A, en el fichero F$. Ahora bien, en muchas ocasiones, por ejemplo en un programa de agenda, se prepara una matriz de datos grande, muy superior al número de datos actual, en previsión de futuras ampliaciones. Si la grabación se realiza tomando como límite el de dimensión de la matriz, resultará que además de la información normal, es almacenada un montón de datos vacios, todos aquellos elementos no asignados por programa y que contienen el vacio por efecto del dimensionamiento (DIM).

Con el fin de almacenar únicamente los datos significativos puede introducirse un IF para detectar la primera cadena vacía e interrumpir el proceso de grabación.

La rutina modificada sería:

```
100 ' RUTINA DE GRABACION PARCIAL DE UNA MATRIZ
115 ' ................. FS9 ...................
120  INPUT "NOMBRE DEL FICHERO";F$
130  OPEN "O",1,F$
140  FOR K=1 TO 100
150 IF A$(k)="" THEN 170 ELSE  WRITE #1,A$(K)
160  NEXT K
170  CLOSE
180 PRINT "Grabacion terminada"
```

El supuesto de esta rutina es una matriz A$, dimensionada previamente como DIM A$(100) y cargada parcialmente de datos.

La carga de datos de la matriz puede realizarse por cualquiera de los métodos expuestos en el capítulo 2 (matrices), por ejemplo añada esta rutina a la anterior para su funcionamiento conjunto.

```
10 ' CARGA DE DATOS EN LA MATRIZ
15 ' ........ FS10 .............
20 DIM A$(100)
30 FOR K=1 TO 100
40 PRINT"Dato";K;
50 INPUT A$(k)
60 IF a$(k)="" THEN 80
```

```
70 NEXT k
80 PRINT"Carga terminada"
90 PRINT"La rutina FS9 procedara a la grabacion"
```

Si se utiliza una carga como la descrita podía aprovecharse el valor de la variable K del bucle como número de datos introducidos, observe que de la 60 se sale con un valor K igual al número de datos mas 1 (K = N + 1), por tanto la rutina FS8 podía modificarse en su línea 140 y 170 en la forma.

```
140 FOR N = 1 TO K—1
170 NEXT N
```

y suprimirse la 150 pues resulta ya innecesaria.

A pesar de lo dicho, el método utilizado en FS9 resulta muy eficaz en programas más complejos, en los que continuas modificaciones del fichero hagan digícil llevar la cuenta del número de datos.

En el caso de datos numéricos, la condición "nada" ("""), debe cambiarse por cero (0), siempre y cuando este número no pertenezca a la lista como valor posible. En caso contrario debe utilizarse un valor singular, por ejemplo —1, —9999, etc., sobre el que tengamos la certeza de su no existencia, e introducirlo como último valor de la lista.

La estructura WHILE también es válida para la grabación de datos. En el caso comentado anteriormente de matrices grandes con datos incompletos, puede establecerse el bucle en la siguiente forma.

```
WHILE A$(K) < > ""      (cadenas)
WHILE A(K) > 0          (números)
```

De esta forma mientras se mantiene la condición los datos serán transferidos al disco. El control se produce en el primer caso por la aparición de la cadena vacía y en el segundo por un número negativo o cero.

7. EXISTENCIA DE UN FICHERO

En todas las rutinas expuestas se hace referencia a un fichero, pero puede ocurrir que dicho fichero no exista en el disco por muy diversas razones: no es el disco apropiado, no se ha tecleado bien el nombre, etc. Todo ello conduce a un mensaje de error «File not found» (fichero no encontrado).

Con el fin de asegurar que el fichero de referencia existe, se dispone de la orden FIND$, válida para cualquier tipo de fichero. Puede probarla incluso como comando directo, teclee

PRINT FIND$ ("nombre.tip")

si el fichero tipificado existe devuelve precisamente su nombre. En la tipificación del fichero pueden incluirse comodines (? *).

Su mayor utilidad viene dada en una aplicación conjunta con IF, veamos por ejemplo como se usaría en la rutina FS7, para ello sólo vamos a introducir una nueva línea, la 35.

```
10 ' RUTINA LECTURA DE UN DATO
15 ' ........ FS11 ..........
20 INPUT "Nombre del fichero ";F$
30 INPUT "Numero de orden del dato a leer";N
35 IF FIND$(F$)<>"" THEN 40 ELSE PRINT"El fichero ";F$;
" no existe";GOTO 20
40 OPEN "I",1,F$
50 FOR k=1 TO N
60 INPUT #1,A$
70 NEXT k
80 CLOSE
90 PRINT "El dato No.";N;" del fichero ";F$;" es ";A$
```

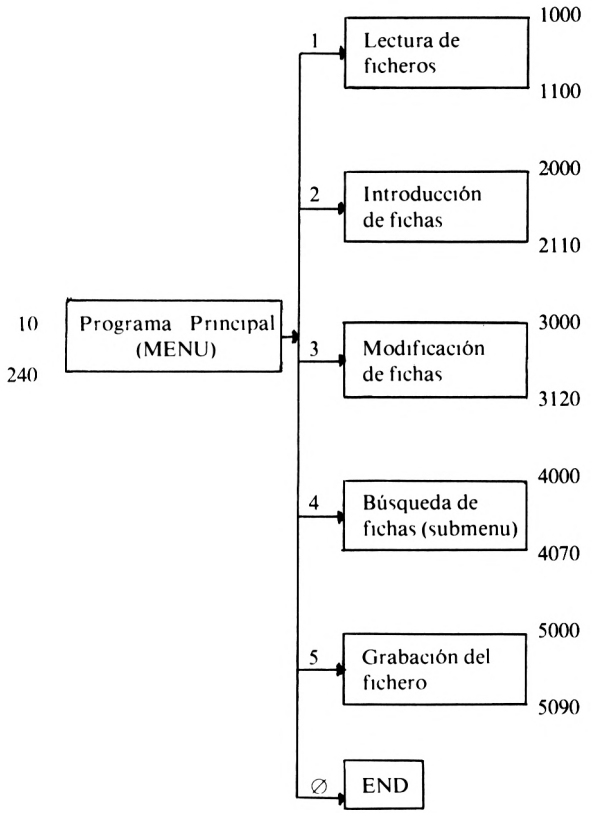
8. EJEMPLO PRACTICO 1. Gestión de biblioteca

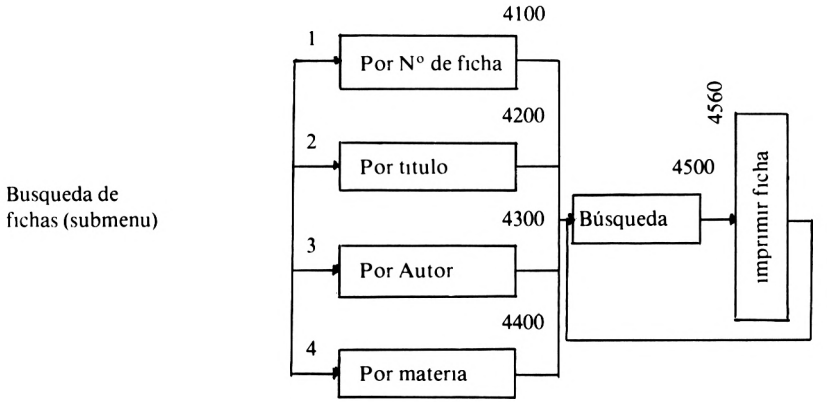
El programa "Gestión de biblioteca" se incluye como ejemplo práctico completo, del manejo de ficheros secuenciales. El objetivo propuesto no es el de construir una compleja base de datos, sino exponer de forma didáctica el esquema básico de una gestión de este tipo de ficheros, por todo ello se ha descuidado, deliberadamente todo lo relativo a una buena presentación, eficaz gestión de menus, etc., buscando en su brevedad, la mejor comprensión de las diversas rutinas que lo componen. Una buena labor por parte del lector sería el perfeccionamiento del programa y adaptación a sus necesidades.

Por su mejor seguimiento, se incluye el organigrama básico del programa y una breve descripción de cada una de las rutinas que la componen.

Organigrama General

El programa general se compone de un pequeño programa "principal" que incluye el dimensionamiento de la matriz de datos y el "Menú" directorio, del que dependen cinco subrutinas operativas, organizadas en la siguiente forma





ORGANIGRAMA BASICO DEL PGM.BIBLIO
→ Flujo principal (los retornos se realizan en sentido contrario)

— **Bases de partida.** El programa se ha concebido con capacidad de gestionar una biblioteca con un máximo de 100 volúmenes, impuesto por el dimensionamiento de la matriz L$ en la línea 50 y tres conceptos (campos): título, autor, materia. Evidentemente estos parámetros pueden ampliarse para ajustarlos a las necesidades reales, el número de libros puede ampliarse modificando el valor 100 y el de conceptos también, aunque esto último obliga a modificar las rutinas ampliándolas en el número de conceptos nuevos introducidos.

Las características de los ficheros secuenciales, permite que el número de caracteres de cada concepto (campo) sea variable.

— **Descripción**

— Programa principal

La línea 50 establece una matriz, dimensionada de acuerdo con las bases de partida expuestas, utilizando la orden OPTION BASE 1 con el fin de utilizar como primer índice de la matriz, el uno. Debe tenerse en cuenta que el primer índice de la matriz es precisamente el que identifica el número de ficha (registro).

La línea 60 define CLS$ como una cadena que incluye las órdenes de borrado de pantalla y cursor al origen, que será empleada posteriormente, mediante PRINT.

Las líneas 100 - 200 constituyen el Menú principal de opciones, sobre el que puede actuarse, como se comentó anteriormente, a fin de mejorar si se desea su forma de presentación.

La línea 205 y 210 permiten el direccionamiento a las distintas subrutinas, indicadas en el organigrama anterior y que se describen a continuación.

— Opción ∅. Fin de programa

Está constituida por sólo dos líneas 230 y 240, cuya única misión es borrar la pantalla y finalizar el programa. Antes de utilizar esta opción, si se han realizado modificaciones de las fichas o aumento, debe procederse a la grabación del fichero.

Si desea la automatización del proceso de cierre puede incluirse una línea 235 GOSUB 5000, de esta forma, el fin de programa producirá una grabación actualizada de datos.

— Rutina 1. Lectura de ficheros.

La línea 1010 realiza una puesta a cero del contador de fichas (K) y de la matriz (L$), ello permite trabajar con varios ficheros, sin que la información sea mezclada.

El bucle 1040-1070 de lectura se ha estructurado mediante un WHILE en la forma de detención por final de fichero, explicada anteriormente, la variable K (línea 1050) es la encargada de contabilizar el número de fichas leidas.

El segundo subíndice de la matriz corresponde a:

Ficha N° = K

Título	L$(K,1)
Autor	L$(K,2)
Materia	L$(K,3)

La línea 1090 produce una pausa hasta tanto no se presione RETURN, INTRO ó ENTER, según el modelo de ordenador. Posteriormente se devuelve el control por la 1100 al programa principal (vuelta al menú)

— Rutina 2. Introducir fichas

La línea 2020 incrementa el contador de fichas. Tras la introducción de los datos, líneas 2030-2050, se produce un "control de bondad" de los mismos, en caso de ser válidos, se ofrece la posibilidad de introducir más fichas, sin tener que pasar por el menú, esta opción es muy útil durante el proceso de creación del banco de datos.

Los controles de bondad y de introducción de nuevos ficheros, utilizan la función UPPER$, con el fin de corregir la diferencia entre letras mayúsculas y minúsculas, que podía dar lugar a errores de interpretación.

— Rutina 3. Modificación de fichas.

Esta rutina se ha elaborado en forma muy simple. Solicita el número de ficha a modificar (3020), imprime los datos actuales (3040) y solicita los nuevos. Observe que han de introducirse los tres datos de forma obligada, una mejora consistiría en posibilitar la corrección de un dato determinado, sin obligar a teclear todos forzosamente, esto puede ser un buen ejercicio para el lector interesado.

— Rutina 4. Búsqueda de fichas.

Esta rutina da origen a 4 subrutinas (ver organigrama básico), cada una de ellas especializada por tipo de búsqueda (campos) y otra común a todas ellas, que es en realidad la estructura encargada de la localización y presentación de la ficha buscada.

Las líneas 4010 a 4050 presentan un menú de búsqueda, para él cabe lo dicho anteriormente para el principal, en cuanto a presentación.

La 4060 se encarga del direccionamiento y 4070 del retorno al menú principal.

Rutina 4-1. Búsqueda por número de ficha.

Es evidentemente el caso más sencillo, conocido el número de ficha, basta con imprimir los tres elementos de la matriz correspondiente. La línea 4110 controla la existencia de la ficha buscada, si el número (NF) es superior a (K) implica que la ficha como tal no existe, o lo que es lo mismo, dicha ficha está vacía.

Se ha añadido la posibilidad de ver todas las fichas seleccionando esta opción y buscando la ficha número cero, que evidentemente no existe pero que sirve como indicador. Para ello se establece un bucle de búsqueda entre los límites "prin" y "fin" cuyos valores serán iguales si se busca una sola ficha o valdrán 1 y K respectivamente si se desa ver todo el fichero.

Rutina 4-2. Búsqueda por Título.

En realidad tanto esta como las dos posteriores se encargan únicamente de establecer el código del campo de búsqueda (1 a 3), es decir, el elemento de la matriz por el que se busca (segundo índice) y la cadena buscada, resolviéndose por la rutina 4.5 la búsqueda propiamente dicha.

Rutina 4-5. Rutina general de búsqueda.

Entre los límites 1 y K, establecidos por el bucle 4510-4530, se realiza la búsqueda del dato propuesto, utilizando el método de "inclusión" como recordará por el capítulo de matrices, este método es mejor que el de la "igualdad", ya que permite buscar por la cadena completa o simplemente por una subcadena contenida. Por ejemplo, si queremos ver los libros de un determinado autor, del que recordamos su apellido pero no el nombre completo, la función INSTR permite su localización, mientras que con el operador de relación (=) esto no sería posible.

Recuerde que la forma general de INSTR es:

INSTR (cadena en la que se busca, subcadena buscada)

y que devuelve el ordinal del primer carácter coincidente de la subcadena, por tanto si vale cero (0), no está contenido y si es distinto de cero si, de esta función se encarga la línea 4520.

Como la impresión de datos, 4560-4620 está organizada como otra subrutina, permite una búsqueda contínua extendida a todo el fichero, aún cuando se haya encontrado una ficha que responda a la condición previa, ello resulta necesario pues permite de esta forma listar por ejemplo todos los libros de la materia Informática y no sólo el primero de ellos.

Rutina 5. Grabación de datos.

Es la operación contraria a la lectura y por tanto de análoga estructura. La apertura es del tipo "O", output, hacia afuera (5020) y la operación de escritura viene dada por WRITE (5040), al igual que la lectura se hacía mediante INPUT.

Observe que tanto la operación de lectura como de escritura, finaliza con el cierre del fichero (1080 y 5070).

```
10 ' ********************** GESTION DE BIBLIOTECA ******************
20 ' **********************  Fichero secuencial   ******************
30 ' **********************       "BIBLIO"        ******************
40 '
50 OPTION BASE 1 DIM L$(100,3)
60 CLS$=CHR$(27)+"E"+CHR$(27)+"H"
100 ' ..  .... .  .............  ................ MENU
110 PRINT CLS$
120 PRINT TAB (22)"M  E  N  U"
130 PRINT TAB (22)"==========="
140 C=20·PRINT:PRINT TAB(C)"1.LEER FICHERO"
150 PRINT·PRINT TAB(C)"2 Introducir ficha"
160 PRINT.PRINT TAB(C)"3 Modificar ficha"
```

```
170 PRINT:PRINT TAB(C)"4.Buscar ficha"
180 PRINT:PRINT TAB(C)"5.GRABAR FICHERO"
190 PRINT:PRINT TAB(C)"0.     F  I  N"
200 PRINT:PRINT:INPUT "SELECCIONAR OPCION ",OP
205 IF OP=0 THEN 230
210 ON OP GOSUB 1000,2000,3000,4000,5000,230
220 GOTO 110
230 PRINT CLS$
240 END
1000 ´ ..................... ...................LEER FICHERO
1010 K=0:ERASE L$:DIM L$(100,3)
1020 PRINT CLS$:INPUT "NOMBRE DEL FICHERO ";F$
1030 OPEN "I",1,F$
1040 WHILE NOT EOF(1)
1050 K=K+1
1060 INPUT #1, L$(K,1),L$(K,2),L$(K,3)
1070 WEND
1080 CLOSE
1090 INPUT "FICHERO CARGADO. PRESIONE <ENTER> PARA CONTINUAR",Z$
1100 RETURN
2000 ´ ..................................... INTRODUCIR FICHAS
2010 PRINT cls$
2020 K=K+1:PRINT"Ficha No.";K
2030 INPUT "Titulo ........... ";L$(K,1)
2040 INPUT "Autor  ........... ";L$(K,2)
2050 INPUT "Materia ......... ";L$(K,3)
2060 PRINT:INPUT "DATOS CORRECTOS (S/N) ";Z$
2070 IF UPPER$(Z$)="S" THEN 2090
2080 IF UPPER$(Z$)="N" THEN 2020
2090 PRINT:INPUT "MAS FICHAS  (S/N) ";Z$
2100 IF UPPER$(Z$)="S" THEN 2010
2110 IF UPPER$(Z$)="N" THEN RETURN ELSE 2090
3000 ´ ..................................... MODIFICAR FICHAS
3010 PRINT CLS$
3020 INPUT "NUMERO DE LA FICHA ";N
3030 PRINT
3040 FOR J=1 TO 3:PRINT L$(N,J):NEXT
3050 PRINT:PRINT"INTRODUCIR NUEVOS DATOS"
3060 INPUT "Titulo ........... ";L$(N,1)
3070 INPUT "Autor  ........... ";L$(N,2)
3080 INPUT "Materia ......... ";L$(K,3)
3090 PRINT:INPUT "DATOS CORRECTOS (S/N) ";Z$
3100 IF UPPER$(Z$)="S" THEN 3120
3110 IF UPPER$(Z$)="N" THEN 3050 ELSE 3090
3120 RETURN
4000 ´ ..................................... BUSCAR FICHAS
4010 PRINT CLS$:PRINT"1. Buscar por numero de ficha"
4020 PRINT"2. Buscar por Titulo"
4030 PRINT"3. Buscar por Autor"
4040 PRINT"4. Buscar por Materia"
4050 PRINT:INPUT "SELECCIONE OPCION ";OP
4060 ON OP GOSUB 4105,4200,4300,4400
4070 RETURN
4100 ´ Busqueda por numero de ficha
4105 INPUT "NUMERO DE FICHA (0 Para ver todas)";NF
4110 IF NF>K THEN PRINT"NO EXISTE ESA FICHA (MAX=";K;")":GOTO 4155
4115 IF NF=0 THEN prin=1:fin=k ELSE prin=nf:fin=nf
4120 FOR ficha=prin TO fin
4125 PRINT CLS$:PRINT"Ficha ";ficha
4130 PRINT "Titulo ....... ";L$(ficha,1)
4135 PRINT "Autor  ..... ";L$(ficha,2)
4140 PRINT "Materia .... ";L$(ficha,3)
```

```
4145 PRINT PRINT PRINT· INPUT "Presione <ENTER> para continuar o S+<ENTER>
alir ",Z$
4150 IF Z$="" THEN NEXT ficha ELSE 4160
4155 INPUT "Presione <ENTER> para volver al menu",Z$
4160 RETURN
4200 ´ Busqueda por Titulo
4210 INPUT "Titulo buscado ",LB$
4220 COD=1
4230 GOTO 4500
4300 ´ Busqueda por Autor
4310 INPUT "Autor buscado ",LB$
4320 COD=2
4330 GOTO 4500
4400 ´ Busqueda por Materia
4410 INPUT "Materia buscada ",LB$
4420 COD=3
4430 GOTO 4500
4500 ´ Rutina general de busqueda
4510 FOR J=1 TO K
4520 IF INSTR(L$(J,COD),LB$)<>0 THEN GOSUB 4560
4530 NEXT J
4540 PRINT PRINT"NO' HAY MAS FICHAS"
4550 PRINT INPUT "Presione <ENTER> para continuar ",z$
4560 PRINT CLS$
4570 PRINT"Ficha   ",J
4580 PRINT "Titulo       ",L$(j,1)
4590 PRINT "Autor       ",L$(j,2)
4600 PRINT "Materia      ",L$(j,3)
4610 PRINT PRINT PRINT INPUT "Presione <ENTER> para continuar ",z$
4620 RETURN
5000 ´                                    GRABAR FICHERO
5010 PRINT CLS$ NPUT "NOMBRE DEL FICHERO ",F$
5020 OPEN "O",1,F$
5030 FOR J=1 TO K
5040 WRITE #1,L$(J,1),L$(J,2),L$(J,3)
5050 NEXT
5060 PRINT"FICHERO "F$,"GRABADO"
5070 CLOSE
5080 INPUT "PRESIONE <ENTER> PARA CONTINUAR",Z$
5090 RETURN
```

9. EJEMPLO PRACTICO 2. Juego educativo de preguntas y respuestas

El segundo ejemplo que se expone de manejo de ficheros secuenciales, corresponde a un sencillo juego educativo, del tipo preguntas y respuestas, que parecen estar de nuevo de moda.

Una vez más el objetivo propuesto es el didáctico y no el de tratar de mejorar el "Trivial". El juego se basa en presentar una pregunta elegida aleatoriamente de las almacenadas en memoria, junto con tres respuestas, tipo test, de las que el jugador debe elegir una. El programa le indica si la respuesta es o no correcta y lleva la contabilidad de aciertos y fallos.

Se han desarrollado dos programas independientes, el primero "GENEPRE", permite la generación de los ficheros de preguntas, que serán utilizados posteriormente por el segundo programa, "PREYRE" y que constituye el juego propiamente dicho.

— Pgm. GENEPRE

Debe introducirse el número de preguntas que contendrá el archivo y el nombre del mismo.

Tras la apertura del archivo, se procede a grabar en primer lugar el número de preguntas y posteriormente, por bloques, la pregunta, las tres respuestas y el número de la respuesta válida.

A pesar de cada vuelta del bucle incluye una orden de escritura, observará que al principio no se ilumina el disco, esto es debido a la existencia de un buffer intermedio, que retrasa dichas órdenes hasta tanto no está lleno, de esta forma se minimiza el número de acceso al disco.

```
10 ' ********* PGM. GENERADOR DE PREGUNTAS Y RESPUESTAS ***********
20 ' ******* CREA LOS FICHEROS UTILIZADOS POR  "PREYRES"  *********
30 ' *********              "GENEPRE"                   ***********
40 '
50 CLS$=CHR$(27)+"E"+CHR$(27)+"H"
60 PRINT CLS$
70 PRINT" ********* PGM. GENERADOR DE PREGUNTAS Y RESPUESTAS ***********"
80 PRINT:INPUT "NUMERO DE PREGUNTAS ";NP
90 PRINT:INPUT "NOMBRE PARA EL FICHERO ";F$
100 OPEN "O",1,F$
110 WRITE #1,NP
120 FOR K=1 TO NP
130 PRINT CLS$
140 PRINT"No.";K
150 INPUT "Pregunta      ",P$
160 INPUT "Respuesta 1: ",R1$
170 INPUT "Respuesta 2· ",R2$
180 INPUT "Respuesta 3: ",R3$
190 INPUT "Cual es la respuesta acertada (1,2 o 3) ",R
200 PRINT:INPUT "DATOS CORRECTOS (S/N) ";Z$
210 IF UPPER$(Z$)="S" THEN 240
220 IF UPPER$(Z$)="N" THEN 130
230 GOTO 200
240 WRITE #1,P$,R1$,R2$,R3$,R
250 NEXT K
260 CLOSE
270 PRINT CLS$;"FIN DE ENTRADA DE PREGUNTAS"
280 PRINT:INPUT "Generar otro fichero (S/N) ";Z$
290 IF UPPER$(Z$)="S" THEN RUN
300 IF UPPER$(Z$)="N" THEN END
310 GOTO 280
```

— Pgm. PREYRE

Dada la simplicidad del mismo, tanto de programación como de uso, resultaría prólija su descripción detallada, por tanto, se destacan a continuación únicamente los detalles más significativos.

90 Antes de proceder a la lectura del fichero indicado, se comprueba su existencia en el disco, tal como se indicó en el apartado 7 de este capítulo.

120 Dado que todos los datos del fichero serán almacenados en memoria principal, se procede al dimensionamiento de matrices que los contengan. Aunque se hubiese podido utilizar una sola matriz bidimensional, como se hizo en el ejemplo anterior, se han utilizado 6 matrices de una dimensión, P$, R1$, R2$ y R3$, para las preguntas y respuestas, R, para los numeros de las respuestas y SOL, para controlar las preguntas que ya han salido, con el fin de que no se repitan.

210 La variable SUMA es el contador del número de preguntas.

220 Dado que el número total de preguntas, almacenado en disco, es recogido por la variable NP, esta línea controla el fin de juego por agotamiento del número de preguntas.

230 Elección aleatoria del número de la pregunta.

350 Esta línea en combinación con la 360 y 420 y subsiguientes controlan el acierto y el fallo. La variable A acumula los aciertos y F, los fallos. Observe como en ambos casos el elemento correspondiente de la matriz SOL, es puesto a uno con el fin de que la 240 pueda controlar el que no se repitan las preguntas.

```
10 ´ *****************   PREGUNTAS Y RESPUESTAS   ******************
20 ´ *****************        "PREYRE"            ******************
30 ´
40 CLS$=CHR$(27)+"E"+CHR$(27)+"H"
50 PRINT CLS$
60 PRINT:INPUT "NOMBRE DEL FICHERO DE PREGUNTAS ";F$
70 ´..................... COMPROBACION Y LECTURA DEL FICHERO
80 ´
90 IF FIND$(F$)="" THEN  PRINT"El fichero ";F$;" no existe":GOTO 60
100 OPEN "I",1,F$
110 INPUT #1,NP
120 OPTION BASE 1:DIM P$(NP),R1$(NP),R2$(NP),R3$(NP),R(NP),SOL(NP)
130 FOR K=1 TO NP
140 INPUT #1,P$(K),R1$(K),R2$(K),R3$(K),R(K)
150 NEXT K
160 CLOSE
170 PRINT:PRINT"FICHERO DE PREGUNTAS   ";F$;"   CARGADO"
180 PRINT:INPUT "PRESIONAR <RETURN> PARA CONTINUAR ";Z$
190 ´............................... BUCLE DE PREGUNTAS
200 ´
210 SUMA=SUMA+1
220 IF SUMA>NP THEN 500
230 X=INT(RND*NP)+1
240 IF SOL(X)=1 THEN 230
250 PRINT CLS$;"ACIERTOS =";A,"FALLOS =";F,,"PREGUNTAS =";SUMA-1
260 PRINT:PRINT TAB(28)"PREGUNTA No.";SUMA
270 PRINT
280 PRINT "Pregunta :    ",P$(X)
290 PRINT "Respuesta 1: ",R1$(X)
300 PRINT "Respuesta 2: ",R2$(X)
310 PRINT "Respuesta 3: ",R3$(X)
320 PRINT:PRINT
330 INPUT "Introduce la respuesta acertada (1,2 o 3) ";R
340 IF R<1 OR R>3 THEN 330
350 IF R=R(X) THEN 360 ELSE 420
360 ´........................  ACIERTOS
370 ´
```

```
380 PRINT:PRINT "ACIERTO !"
390 A=A+1:SOL(X)=1
400 PRINT:PRINT:PRINT:
410 GOTO 480
420 '........................  FALLOS
430 '
440 PRINT:PRINT"FALLO"
450 F=F+1:SOL(X)=1
460 PRINT:PRINT"LA RESPUESTA ACERTADA ES LA No. ";R(X)
470 PRINT:PRINT:PRINT
480 PRINT:INPUT "PRESIONAR <RETURN> PARA CONTINUAR ";Z$
490 GOTO 210
500 ' ....................  FIN DE PROGRAMA
510 PRINT CLS$:PRINT:PRINT TAB(27)"R E S U L T A D O S":PRINT
520 PRINT TAB(26)"=====================":PRINT:PRINT
530 PRINT "ACIERTOS =";A,,"FALLOS =";F,,"PREGUNTAS =";SUMA-1
540 PRINT:PRINT:PRINT:PRINT:
550 END
```

Utilice GENEPRE para preparar las preguntas y las respuestas. Cada pregunta ha de componerse de:

Pregunta
Respuesta 1
Respuesta 2
Respuesta 3
Nº de respuesta válido (1 a 3)

Por ejemplo

Pregunta: ¿Quién descubrió América?
Respuesta 1: Julio Iglesias
Respuesta 2: Cristóbal Colón
Respuesta 3: Magallanes
Nº respuesta: 2

Mediante este programa puede preparar tantos ficheros como desee, p.e. "GEOGRAF", "HISTORIA", etc., que posteriormente puede utilizar mediante RUN "PREYRE", el uso de este programa es autoexplicativo, siga las instrucciones de pantalla.

CAPITULO 6

FICHEROS DE ACCESO DIRECTO

1. INTRODUCCION

2. CONCEPTOS PREVIOS

3. FORMATO DE REGISTRO

4. ASIGNACION DE LAS VARIABLES DE CAMPO

5. APERTURA DE FICHEROS

6. LECTURA Y ESCRITURA DE FICHEROS

7. MODIFICACION DE ARCHIVOS

8. EJEMPLO PRACTICO 1
   Agenda de direcciones

9. EJEMPLO PRACTICO 2
   Gestión de videoclubs

6. Ficheros de acceso directo

1. INTRODUCCION

Hemos visto en el capítulo anterior cómo en los ficheros secuenciales —para leer un registro— se ha de proceder a la lectura de todos los anteriores; y cómo —para corregir datos— hemos de cargar todo el fichero en memoria, modificarlo y grabar el nuevo fichero.

Todos estos "inconvenientes" se subsanan con los ficheros de acceso directo, tambien conocidos como aleatorios. En éstos, el acceso a un dato o grupo de datos es inmediato y directo, sin pasar por los anteriores, tanto para lectura como para grabación, de tal forma que la corrección de un registro no exige la manipulación del resto ni la alteración del fichero en el disco, tomado como conjunto.

Por su propia naturaleza los secuenciales van íntimamente unidos a las matrices, mientras los aleatorios no, puesto que la lectura o grabación de un registro no requiere este tipo de estructura, sino variables simples.

2. CONCEPTOS PREVIOS

El ejemplo más claro del significado del almacenamiento de acceso directo, nos lo ofrece el propio sistema de gestión del disco.

Cuando formateamos un disco, vemos que éste se divide en 40 pistas (círculos concéntricos) y que cada pista se divide a su vez en 9 partes iguales, que reciben el nombre de sectores.

La capacidad de un sector es de 512 bytes, por lo que, realizando una pequeña operación (40 . 9 . 512 = 184.320), llegamos a la conclusión de que cada cara de un disco dispone de 184.320 bytes; lo cual supone 180K de capacidad máxima (184.320/1.024 = 180K).

Cuando se graba un programa, se incluyen en el directorio —además del nombre— la pista y sector donde comienza y donde acaba. Tenga en cuenta que no tienen por qué ser contiguos y, de hecho, lo normal es que estén separados, de manera que un fichero se encuentre repartido por varios sectores del disco.

Al recibir una orden de lectura, se accede, en primer lugar, al directorio y, si existe el fichero, se recoge la información sobre la forma en que dicho fichero está repartido por el disco. Esto constituye un acceso directo, puesto que no hay que leer ficheros anteriores para acceder al buscado.

La antítesis de lo expuesto es la cinta —almacenamiento secuencial— donde para leer un fichero (programa) hay que comenzar desde el principio. Por supuesto que Vd. puede contar las vueltas y saber más o menos donde comienza, pero la maniobra manual de hacer avanzar la cinta sigue constituyendo una búsqueda secuencial (¡y encima "a mano"!).

Cuando se crea un fichero de acceso directo también se crea un índice, que permite localizar cualquier registro; de manera que existe una analogía entre ficheros de acceso directo y disco en cuanto a la forma de guardar la información.

	Disco	*Fichero Acceso Directo*
Directorio de	Ficheros	Registros
Identificación mediante	Nombre del fichero	Número del registro
Organización	Sectores de 512 bytes	Registros de igual longitud

La identificación de un fichero viene dada por su nombre, mientras que la de un registro viene dada por su número de orden. El conocer su número de orden es importante, pues permite el acceso directo propiamente dicho. Esto no quiere decir que sea estrictamente necesario, pues siempre podemos buscar un elemento recurriendo a las técnicas utilizadas en los de acceso secuencial, pero evidentemente perdemos su mayor ventaja.

Una particularidad de estos ficheros es todo lo relativo a la longitud de los registros. En los secuenciales podíamos tener la información almacenada, una tras otra, con longitudes distintas. Por ejemplo:

"LUIS","JOSE MANUEL","PEDRO"
4 11 5

Sin embargo, los registros de un fichero de acceso directo han de tener igual longitud. Esto se consigue llenando de blancos los espacios sobrantes, como se verá más adelante. Volviendo al ejemplo anterior y representando con un "_" al espacio, los registros anteriores, definidos como de longitud 11 quedarían:

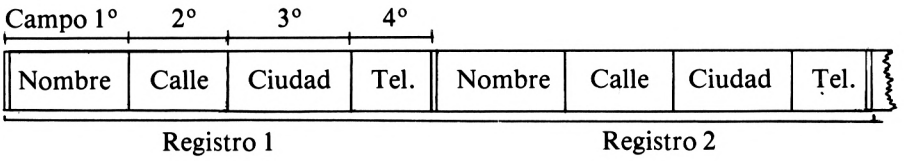
"LUIS _ _ _ _ _ _ _", JOSE_MANUEL","PEDRO _ _ _ _ _ _ _"
11 11 11

La necesidad de que los registros tengan la misma longitud es ovbia, es la única manera de saber el ordenador dónde empieza el siguiente.

Lo primero que salta a la vista es que este tipo de estructura desaprovecha espacio. Más adelante, veremos cómo se crean estos ficheros y se dimensionan sus registros, a fin de optimizar —en lo posible— el tamaño del fichero. Pero antes vamos a matizar qué es un registro y de qué está compuesto.

Un registro no es como en los ficheros secuenciales —un hueco donde introducir un dato— sino que contiene subapartados denominados campos, y en cada campo puede introducirse un dato.

La forma más sencilla de comprender esta estructura es imaginar que un registro es como una tarjeta de visita, que contiene nombre, dirección, ciudad, teléfono, etc. Pues bien, cada uno de estos apartados es un campo, y además, estos campos guardan un cierto orden establecido: primero el nombre, segundo la calle, etc. De esta forma, cada registro contendrá los siguientes campos:



Un registro puede contener perfectamente un solo campo o varios sin limitación.

3. FORMATOS DE REGISTRO

Antes de trabajar con ficheros aleatorios, debe fijarse la longitud total de los registros y de cada uno de los campos que lo componen. La apertura de un fichero aleatorio implica la reserva de un espacio estandar de memoria, de 128 bytes si no se indica explícitamente otro valor (ver MEMORY), dicha reserva se conoce como "tampón de registros".

El formato de registro, que constituye el número de campos y su distribución en el registro debe colocarse en el tampón, antes de realizar grabaciones en el fichero. Es en definitiva una plantilla de como ha de almacenarse la información.

Para ver prácticamente como se utilizan dichos formatos o plantillas, vamos a considerar un ejemplo que será desarrollado más tarde en el programa "AGENDIR", al final del capítulo.

Se trata de construir un fichero de direcciones (agenda), cuyos registros han de contener:

Nombre, dirección, población, provincia, código postal y teléfono, estos registros estarán formados por seis (6) campos, que serán identificados por las siguientes variables, respectivamente:

NUM$, CL$, POB$, PRO$, CP$, TEL$
(el nombre de las variables es establecido libremente por el programador)

La longitud establecida en este ejemplo, para cada campo, es la indicada en el siguiente cuadro

Variable de campo	*Ejemplo de contenido*	**Longitud de campo**	*Caracteres* del *ejemplo*
NOM$	Editorial ra-ma	35	15
CL$	Carretera de Canillas - 144	40	25
POB$	Madrid	15	6
PRO$	Madrid	12	6
CP$	28.043	6	6
TEL$	91 - 200 97 46 - 47	15	13

Se ha previsto pues para cada campo una longitud máxima, es decir, un número de caracteres como límite, pero ello no impide que puedan almacenarse menores, como se ven en la última columna del cuadro.

La longitud del registro, será la suma de la de todos los campos que lo componen, en el ejemplo

$$35 + 40 + 15 + 12 + 6 + 15 = 123$$

Como 123 es menor que la longitud del tampón estandar no es necesario modificarlo con MEMORY.

Tanto las variables identificativas de los campos, que denominaremos "variables de campo", como las distintas longitudes y por extensión la total, se definen mediante la orden FIELD (campo). Su síntaxis general es:

FIELD n° fichero, long. campo 1 AS var.1, Long. campo 2 AS var.2, ...

Siguiendo con el ejemplo y suponiendo que el número de fichero será uno, la orden de formato de registros será

FIELD 1, 35 AS NOM$, 40 AS CL$, 15 AS POB$, 12 AS PRO$, 6 AS CP$,
15 AS TEL$

Para un mismo fichero y en caso necesario, pueden definirse varios formatos de registro.

Las variables de campo, una vez definidas por FIELD, son distintas del resto de variables alfanuméricas, en el sentido de que su longitud es fija y en que su asignación no se realiza con la sentencia LET, sino con alguna de las sentencias que se indican a continuación.

4. ASIGNACION DE LAS VARIABLES DE CAMPO

Dado que la longitud de un campo es fija y como consecuencia lo será la variable de campo correspondiente, ocurre generalmente que al utilizar datos reales, sus longitudes son variables, en el ejemplo del apartado anterior (cuadro), el primer campo tiene una longitud de 35, mientras que el nombre sólo ocupa 15 caracteres, los 20 ca-

racteres de diferencia habrán de obtenerse añadiendo espacios. Esto puede hacerse de dos formas: añadiendo blancos por la izquierda o por la derecha de la cadena.

Mallard BASIC dispone de dos órdenes de asignación de variables de campo, que realizan de forma automática el relleno de blancos. La forma general de estas órdenes es

$$\text{LSET var. campo} = \begin{cases} \text{"Expresión literal"} \\ \text{variable alfanumérica} \end{cases}$$

$$\text{RSET var. campo} = \begin{cases} \text{"Expresión literal"} \\ \text{variable alfanumérica} \end{cases}$$

Como quizá haya podido adivinar L y R son las iniciales de las funciones de fragmentación LEFT$ y RIGHT$, de manera que LSET asigna a la variable de campo el literal correspondiente, alineándolo por la izquierda, mientras RSET lo hace por la derecha.

Veamos un sencillo ejemplo, sea A$ la variable de campo definida por

FIELD 1, 6 AS A$

Si suponemos que Z$ = "HOLA", la sentencia de asignación

LSET A$ = Z$ ⇒ A$ = "HOLA__"

mientras

RSET A$ = Z$ ⇒ A$ = "__HOLA"

representando los espacios por el símbolo (_).

Además de estas dos sentencias de asignación, Mallard BASIC dispone de otra, ya conocida pues se utiliza para fragmentar cadenas. Su forma general es

$$\text{MID\$ (var. campo, origen, longitud)} = \begin{cases} \text{"expresión literal"} \\ \text{variable alfanumérica} \end{cases}$$

Permite tomar del literal una subcadena de la longitud indicada y la asigna a la variable de campo, a partir de la posición dada por el origen.

Siguiendo con el ejemplo último (RSET), si aplicamos posteriormente

MID$ (A$, 1, 2) = B$ siendo B$ = "PINO"

se obtendrá D$ = "PIHOLA".

Todo lo expuesto se refiere al campo alfanumérico, si se trata de almacenar números, también pueden utilizarse las tres asignaciones descritas, realizando la transformación de número a literal numérico mediante la función STR$, por ejemplo si

tiene que almacenarse el número 12.340 contenido en la variable N, en la variable de campo A$, por la izquierda puede utilizarse

LSET A$ = STR$(N) siendo N = 12340

Esta forma aunque válida, resulta ineficaz desde el punto de vista de ocupación, ya que el número como tal ocupa menos bytes que el literal correspondiente. Para ello se dispone de tres funciones que transforman números, a efectos de almacenamiento en archivo, en cadenas más compactas que las obtenidas por STR$, aunque estas cadenas no pueden ser representadas gráficamente.

MKD$ (número)	Convierte un número de doble precisión en cadena de 8 caracteres.
MKDS$ (número)	Convierte un número de precisión sencilla en cadena de 8 caracteres.
MKI$ (número)	Convierte un número entero en una cadena de 2 caracteres.

De la misma forma que la función STR$ tiene una inversa, VAL, existen otras tantas inversas para las anteriormente expuestas y son respectivamente, CVD, CVS y CVI, que convierte cadenas en números.

CVD (cadena)	Convierte una cadena de 8 bytes en un número de doble precisión.
CVS (cadena)	Convierte una cadena de 4 bytes en un número de simple precisión.
CVI (cadena)	Convierte una cadena de 2 bytes en un número entero.

Volviendo al ejemplo anterior en que se utilizó STR$, la conversión en cadena de N podía ser

LSET A$ = MKI$ (N)

Se utiliza MKI$, por ser N = 12340 (número entero). De esta forma el dato puede proceder a su grabación en archivo, en la forma que se verá más adelante. Una posterior lectura obligaría a la transformación apropiada para poder ser usado como tal valor numérico, la función sería CVI, por ejemplo

N = CVI (A$)

Aunque en el ejemplo se utiliza LSET como sentencia de asignación, recuerde que también dispone de RSET y MID$.

5. APERTURA DE FICHEROS

La orden de apertura de ficheros aleatorios, es la misma que la utilizada anteriormente para los ficheros secuenciales, únicamente se diferencia en el tipo de lectura,

en los secuenciales podía ser "O" para escribir e "I" para leer, en los aleatorios (acceso directo) solo existe el tipo "R" (R-Random/aleatorio), tanto si es para lectura como para escritura, no hay pues distinción.

La forma general es:

OPEN "R", número fichero, nombre fichero

Una función muy útil antes de la apertura de un fichero, es la realizada por FIND$, descrita en el apartado 7 del capítulo anterior, y cuyo uso es muy recomendable, a fin de no producir interrupciones en la marcha del programa por la no existencia de un fichero determinado.

NOTA Si se abre un fichero y este no existe, la orden OPEN lo crea

6. LECTURA Y ESCRITURA DE REGISTROS

Una vez abierto un fichero de acceso directo y definida su plantilla y variables de campo, las órdenes de lectura y escritura son muy simples.

— **Lectura.** Se utiliza la orden GET en la forma:

GET número fichero, número registro

Si se omite el segundo parámetro (número registro), la lectura se realiza secuencialmente, es decir el registro leído es el siguiente al último leído con GET. En cualquier caso la operación de lectura afecta a un registro completo y no a un determinado campo, sin embargo el contenido de cada campo es almacenado en cada una de las variables de campo definidas por FIELD, así en el ejemplo propuesto en el apartado 3 de este capítulo y suponiendo que se almacenó como registro uno y que el fichero es también el número uno, la orden

GET 1,1,

provocaría una asignación de los datos de dicho ejemplo, a las variables de campo que se indicaron, es decir

```
NOM$ = "Editorial ra-ma"
CL$  = "carretera de Canillas 144"
.............................................
TEL$ = 91 - 200 97 46 - 47
```

En los ejemplos prácticos incluidos al final del capítulo, puede observarse con detalle dos casos pormenorizados de lectura.

— **Escritura.** La operación ha de ser análoga a la lectura y por tanto la forma de PUT, que es la orden de grabación, será:

PUT número de fichero, número de registro

En este caso son las variables de campo las que producen el volcado de su información sobre el disco, en el fichero y registro indicado.

La omisión del segundo parámetro producirá la grabación secuencial de los registros. Esta forma suele ser usada únicamente cuando se trata de inicializar el fichero y los datos están contenidos en una matriz, la única ventaja es el tiempo, ya que al realizar la grabación de registros uno tras otro, no necesita calcular la posición de escritura de cada uno de ellos.

7. MODIFICACION DE ARCHIVOS

El acceso directo a un registro permite que las modificaciones puedan realizarse sobre el propio registro, sin afectar al resto, al contrario de los secuenciales en que debe cargarse todo el fichero, modificarse en memoria y volverse a grabar completo. El trabajo de lectura y escritura en los de acceso directo es normalmente sobre un registro, salvo la excepción hecha de su posibilidad de trabajo secuencial y aún en este caso su manejo de campos los hace más potentes.

EJEMPLO PRACTICO 1. Agenda de direcciones

Como resumen de las explicaciones anteriores sobre el uso y manejo de ficheros de acceso directo, se incluye un programa completo, que permite llevar una agenda personal y que con pequeños retoques puede adaptar a sus necesidades particulares. Una vez más se han tratado de unir lo práctico y lo didáctico, habiéndose cuidado su longitud, de forma que no resulte muy penosa su copia, aunque puede obviarla mediante el disco ofertado en la nota al pie del índice general.

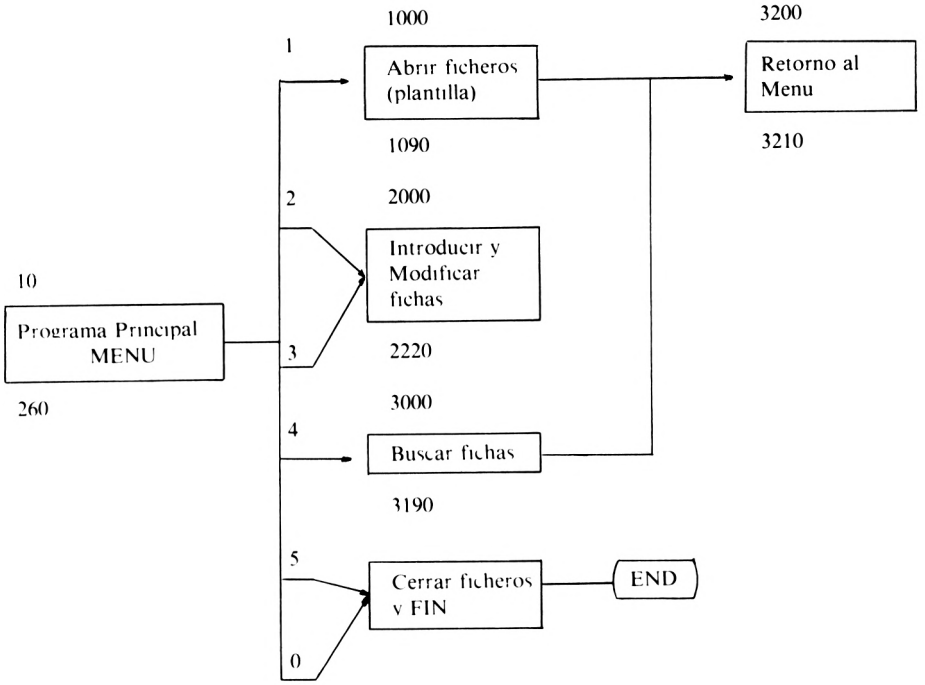
Organigrama general

Se ha utilizado la técnica de un pequeño programa principal, compuesto principalmente por un Menú de opciones y una serie de rutinas dependientes y direccionadas por él. Su esquema básico es el de la página siguiente.

De las 6 opciones del Menú la 2ª y 3ª se han resuelto juntas, así como la 5° y la 0°, ya que como se verá más adelante simplifican el programa.

Bases de partida

El programa permite gestionar una agenda de direcciones con un número de fichas limitado por la capacidad del disco, seguramente muy superior a las necesidades corrientes. Cada ficha contiene nombre, dirección, población, provincia, código postal y teléfono, datos que aún siendo bastante completos, pueden aumentarse introduciendo pequeñas variaciones en el programa, no dude en realizarlas personalizando así "su agenda".



Descripción

— Programa principal. Menú.

La línea 60 establece una variable que contiene los caracteres de borrado de la pantalla, a la que denominamos CLS$, por su parecido con la orden CLS de otras versiones BASIC y que hemos utilizado en otros programas. La 70 asigna a S$ una cadena de 66 asteriscos, que será utilizada para la presentación del menú en pantalla.

Las líneas 100 a 230 presentan el menú de opciones, que se indica en el organigrama básico.

Las líneas 240 y 250 realizan la selección y direccionamiento a la subrutina correspondiente. La repetición de los números 2000 y 4000 corresponde al comentario hecho anteriormente sobre la unificación de las opciones 2ª /3ª y 5ª/0.

La 260 produce el retorno al Menú, tras la vuelta de las subrutinas.

Las líneas 270 y 280 son en realidad innecesarias y no se han suprimido por si el lector prefiere que la opción 0 del menú, en vez de direccionarse conjuntamente con la 5ª, se haga de forma independiente, precisamente a 270, cambiando la última línea (4070) por un RETURN en vez de END, ya que como ve hay dos finales cosa poco ortodoxa.

— Rutina 1. Abrir fichero.

El programa maneja dos ficheros, el etiquetado como 1 (1030) es de acceso directo y

contiene las fichas (registros) de direcciones, el fichero número 2, es de tipo secuencial y contiene un solo dato, el número total de fichas del fichero anterior, tiene como nombre fijo, "NUMFICHA", mientras que el del aleatorio viene dado por INPUT (1020).

El segundo fichero podía evidentemente obviarse, guardando el número de fichas en el de acceso directo, construyendo una plantilla especial y utilizando el primer registro. Si se utiliza esta forma el número de ficha sería igual al de registro más uno, ya que el registro número 1 estaría ocupado por el dato singular mencionado.

Se ha preferido la utilización de dos con el fin de trabajar simultáneamente con más de un fichero e incluso de tipos diferentes.

La línea 1030 es la encargada de la apertura del fichero aleatorio, para el que se fija como número de fichero el uno. Si el fichero de direcciones va a recibir un nombre fijo, que es lo lógico puede sustituirse el INPUT de 1020 por un LET, por ejemplo

```
1020 LET F$ = "AGENDA"
```

Las líneas 1050 y 1060 realizan una función de apertura del fichero secuencial, del tipo "O" ó "I", dependiendo de su existencia previa o no. La primera vez que se utiliza el programa, el fichero NUMFICHA no existe, la función FIND$ nos devolverá la cadena vacia y al verificarse la condición de 1050, se procede a la apertura para escribir, se escribre el valor 0 (ninguna ficha) y se cierra el fichero, etiquetado como 2. Si la condición de 1050 no se verifica, ello implica la existencia del fichero NUMFICHA y como consecuencia se procede a la apertura para leer, lectura del número de fichas contenido en él y posterior cierre.

La línea 1040 establece la plantilla y variables de campo, que corresponden al ejemplo propuesto al comienzo del capítulo. La 1070 procede al cierre de este fichero aleatorio.

Observe que el CLOSE 2 de la línea 1050 y el de 1060, así como el CLOSE 1 de 1070, podrían sustituirse por un solo CLOSE, sin número, precisamente ubicado en 1070. La única razón de no hacerlo, es quizá la de dar pie a este comentario, analizando las distintas posibilidades de la orden de cierre.

La 1080 produce un comentario, señalando el nombre del fichero de datos y el número de fichas que contiene, este último dato es importante, pues permitirá añadir nuevas fichas introduciendo como número de registro valores sucesivos más altos del indicado por esta instrucción.

En 1090 se transfiere el control a la rutina 3200, que espera la pulsación de RETURN, para retornar al Menú.

Rutina 2 y 3. Introducir y modificar fichas.

Con el fin de simplificar se ha resuelto la modificación de fichas, como si se tratase de una nueva, es decir, modificar uno o varios datos de un registro, implica teclear todos los datos del mismo nuevamente. Aún no siendo una práctica aconsejable, ha permitido una mayor brevedad y claridad al programa, objetivos propuestos como

prioritarios. Un buen ejercicio por parte del lector puede ser el desdoblamiento de estas rutinas, dejando la existente como Introducción de fichas y elaborando una nueva que permita modificar un campo, sin tener que teclar todos, en caso de hacerlo no olvide modificar la línea 250 de direccionamiento.

La línea 2010 informa sobre cual es la última ficha que se introdujo, o lo que es lo mismo, de cuantas fichas se compone el fichero. La 2020 pide el número de ficha a modificar o introducir.

La 2040 controla si se va a realizar una modificación o se pretende introducir una nueva. Si el numero de ficha es mayor que el de fichas existentes, se trata de una ampliación y por tanto se produce un cambio de la variable UFICHA que pasa a tomar el valor de ficha, siendo

FICHA variable que recoge el número de ficha a modificar o introducir

UFICHA variable que recoge el número total de fichas existentes y que fué leída del fichero NUMFICHA

Si FICHA > UFICHA → UFICHA = FICHA

Por ejemplo, el fichero contenía 16 fichas y en 2010 se dió como ficha a modificar 17, ello implica una ampliación, por lo que se realizará el cambio previsto por 2040.

De 2060 a 2110 se piden los datos de cada uno de los campos, almacenándose en variables auxiliares, distintas de las variables de campo. Si los datos introducidos son correctos (2120 y 2130), se procede a la asignación de las variables de campo, mediante alineación por la izquierda (LSET, líneas 2140/2190).

La 2200 produce la grabación en el registro correspondiente (variable FICHA dada por el INPUT de 2020) y la 2210 el cierre del fichero que se abrió en 2030. Se ha utilizado CLOSE sin parámetros, aún cuando pudo utilizarse CLOSE 1.

La 2220 produce el retorno al Menú.

Rutina 4. Búsqueda de fichas.

La búsqueda se ha establecido en base a la inclusión y únicamente por el nombre. En ejercicios anteriores (ficheros secuenciales) tiene búsquedas más completas.

Tras solicitar el nombre en 3030, que no tiene porque ser completo, la 3050 produce un bucle de rastreo, cuando no se verifica la condición de la 3070, es que se ha encontrado un registro que corresponde al buscado y como consecuencia se imprime su contenido, en caso contrario continúa la búsqueda.

Dado que en una agenda puede buscarse un nombre, p.e. "PEPE" y existir varios, la línea 3160 transfiere el control a 3220 para decidir entre continuar la búsqueda o salir (línea 3240/3190).

Puede ocurrir que el nombre buscado no exista y por tanto se produzca el agotamiento del bucle 3050, por ello se introduce tras el NEXT de 3170 un mensaje al efecto.

Las líneas 3040 y 3190 producen la apertura y cierre respectivamente del fichero de datos.

Rutinas 5 y 0. Cierre de ficheros y fin de la sesión de trabajo.

Dado que el fichero de datos ha sido abierto y cerrado por las distintas rutinas anteriores solo queda en esta operación, la introducción en el fichero del número de fichas, del último valor de la variable UFICHA, por si esta ha tenido modificaciones.

La línea 4020 se encarga de abrir, escribir y cerrar el fichero. La 4030 es redundante y esto además, se incluye únicamente como aviso útil en programación, sobre todo durante el proceso de creación y puesta a punto de un programa, en que es conveniente que el fin de tarea produzca un cierre de todos los ficheros, por si alguno pudo quedar abierto, es una simple medida de seguridad, tenga en cuenta que un CLOSE de más nunca está de más y no produce ningún tipo de error.

Las líneas 4040 y 4070, producen unos mensajes previos al fin de programa. Recuerde lo comentado respecto a un auténtico cierre con el epígrafe del programa principal.

```
10 ´ ******************* AGENDA DE DIRECCIONES  ******************
20 ´ *******************   Fichero aleatorio    ******************
30 ´ *******************      "AGENDIR"         ******************
40 ´
60 CLS$=CHR$(27)+"E"+CHR$(27)+"H"
70 s$=STRING$(66,"*")
100 ´                                              MENU
110 PRINT CLS$
120 PRINT "**********************    A  G  E  N  D  A    ********************"
130 PRINT S$
140 PRINT
150 PRINT TAB (30)"M  E  N  U"
160 PRINT TAB (30)"==========="
170 C=28.PRINT PRINT TAB(C)"1 ABRIR FICHERO"
180 PRINT PRINT TAB(C)"2 Introducir fichas"
190 PRINT PRINT TAB(C)"3 Modificar ficha"
200 PRINT PRINT TAB(C)"4 Buscar ficha"
210 PRINT PRINT TAB(C)"5 CERRAR FICHERO"
220 PRINT PRINT TAB(C)"6      F  I  N"
230 PRINT PRINT S$
240 PRINT PRINT INPUT "SELECCIONAR OPCION ",OP
250 ON OP GOSUB 1000,2000,2000,3000,4000,5000
260 GOTO 110
270 PRINT CLS$
280 END
1000 ´                                ABRIR FICHERO
1010 PRINT cls$
1020 INPUT "Nombre del fichero ",F$
1030 OPEN "R",1,F$
1040 FIELD 1,35 AS NOM$,40 AS CL$,15 AS POB$,12 AS PRO$,6 AS CP$,15 AS TEL$
1050 IF FIND$("NUMFICHA")="" THEN OPEN "O",2,"NUMFICHA" WRITE #2,UFICHA CLOSE 2
GOTO 1070
1060 OPEN "I",2,"NUMFICHA" INPUT #2,UFICHA CLOSE 2
1070 CLOSE 1
1080 PRINT PRINT PRINT"FICHERO  ",F$,"  VERIFICADO  CONTIENE ",UFICHA,"  FICHAS
1090 PRINT GOTO 3200
2000 ´                          INTRODUCIR O MODIFICAR FICHA
2010 PRINT CLS$ PRINT"ULTIMA FICHA ",UFICHA
2020 PRINT INPUT "NUMERO DE FICHA ",FICHA
2030 OPEN "R",1,F$
2040 IF FICHA>UFICHA THEN UFICHA=FICHA
```

```
2050 PRINT CLS$ PRINT"FICHA",FICHA
2060 INPUT "Nombre                ",N$
2070 INPUT "Domicilio             ",D$
2080 INPUT "Poblacion             ",P$
2090 INPUT "Provincia             ",PR$
2100 INPUT "Codigo Postal         ",C$
2110 INPUT "Telefono              ",T$
2120 PRINT INPUT "datos correctos (S/N) ",Z$
2130 IF UPPER$(Z$)<>"S" THEN 2050
2140 LSET NOM$=N$
2150 LSET CL$=D$
2160 LSET POB$=P$
2170 LSET PRO$=PR$
2180 LSET CP$=C$
2190 LSET TEL$=T$
2200 PUT 1,FICHA
2210 CLOSE
2220 RETURN
3000 '                                    BUSCAR FICHA
3010 PRINT CLS$,"BUSQUEDA DE FICHAS"
3020 PRINT"=================="
3030 INPUT "NOMBRE ",N$
3040 OPEN "R",1,F$
3050 FOR K=1 TO UFICHA
3060 GET 1,K
3070 IF INSTR(NOM$,N$)=0 THEN 3170
3080 PRINT CLS$
3090 PRINT"Nombre          ",NOM$
3100 PRINT"Calle           ",CL$
3110 PRINT"Poblacion       ",POB$
3120 PRINT"Provincia       ",PRO$
3130 PRINT"Codigo Postal   ",CP$
3140 PRINT"Telefono        ",TEL$
3150 PRINT PRINT PRINT PRINT
3160 GOTO 3220
3170 NEXT K
3180 PRINT PRINT"NO HAY MAS FICHAS"
3190 CLOSE
3200 INPUT "PRESIONE <RETURN> PARA CONTINUAR",Z$
3210 RETURN
3220 INPUT "INTRODUCIR   C=Continuar S=Salir ",Z$
3230 IF UPPER$(Z$)="C" THEN 3170
3240 IF UPPER$(Z$)="S" THEN 3190
4000 '                    CERRAR FICHERO Y FIN PROGRAMA
4010 PRINT CLS$
4020 OPEN "O",2,"NUMFICHA" WRITE #2,UFICHA CLOSE 2
4030 CLOSE
4040 PRINT"F I C H E R O S    C E R R A D O S"
4050 PRINT PRINT"NUMERO DE FICHAS ALMACENADAS",UFICHA
4060 FOR delay=1 TO 3000 NEXT
4070 RETURN
5000 '                                    FIN DE PROGRAMA
5010 PRINT CLS$ PRINT"ANTES DE DAR POR TERMINADO EL PROGRAMA DEBE CERRAR LOS FI
HEROS UTILIZANDO LA"
5020 PRINT"OPCION 5" INPUT "Desea finalizar (S/N)",Z$
5030 IF UPPER$(Z$)<>"S" THEN RETURN
5040 PRINT CLS$ PRINT"     F I N   D E    P R O G R A M A"
5050 END
```

9. EJEMPLO PRACTICO 2. Gestión de videoclubs.

Este programa, algo más largo que el anterior, es una aproximación a la gestión de un videoclub, contiene los ingredientes básicos, si bien una vez más, se ha reducido al mínimo, buscando una mejor comprensión de forma que el lector interesado, pueda desarrollar sobre su base, programas más complejos.

Se ha mantenido la estructura básica del programa anterior, un fichero aleatorio donde se almacenan los datos personales de los socios y las películas alquiladas ("SOCIOS") y otro secuencial que contiene únicamente el número de socios del club ("NSOCIOS").

El archivo de alquileres no es histórico sino temporal, es decir, los datos son conservados únicamente entre la fecha de alquiler y la de devolución, tras lo cual son destruidos o modificados por nuevo alquiler. No tendrá dificultad, si lo desea en añadir un tercer fichero a este fin, es decir un fichero histórico, del que posteriormente puede sacar información estadística, etc. Por último las películas son manipuladas por su número de identificación y no por título, por lo que quizá fuese conveniente añadir un archivo de títulos. Todas estas posibilidades son muy propias de un programa profesional e implicaría el manejo de una unidad más de disco, puesto que la capacidad de almacenamiento de uno solo es reducida para el caso.

A continuación se describe someramente el programa, habida cuenta del grado de afinidad con el anterior.

— Programa principal. Menú.

Como todo alquiler o devolución va unido a una fecha, el arranque diario del programa, implica la introducción de la fecha en curso (línea 90). Las líneas 110 a 130 en combinación con la 40 y 70 permite la expresión de la fecha en la forma más habitual, a fin de realizar una comprobación de la bondad de la misma, antes de ser asumida por el programa, para su posterior utilización.

El formato de fecha, manipulado por el pgm. y en el cual debe introducirse es: DDMMAA, es decir día con dos dígitos, mes con dos dígitos y año por sus dos últimos dígitos, p.e., el 2 de Mayo de 1987, se introducirá como 020587, la confirmación de fechas se pide sobre el formato normal con expresión literal del mes y año con cuatro dígitos (válido hasta el 2000).

Abrir fichero

Esta rutina es idéntica a la del programa anterior. La plantilla y variables de campo utilizadas es la siguiente

Variable de campo	*Longitud de campo*	*Contenido*
NOM$	25	Nombre del socio
CL$	30	Domicilio
TEL$	7	Teléfono
SIS$	1	Sistema (V-VHS, B-BETA, D-200)
FE$	6	Fecha
P$(1)	5	Película 1
P$(2)	5	Película 2
P$(3)	5	Película 3
P$(4)	5	Película 4
P$(5)	5	Película 5

Como puede verse la fecha se almacena con el formato de entrada descrito y el número máximo de películas permitido es cinco, identificadas por un número máximo de 5 caracteres. Estas premisas, son fácilmente variables y/o ampliables.

Alta de nuevo socio

La rutina 2000 a 2210 permite la introducción de los datos de afiliación de un nuevo socio o bien la modificación de los mismos, tal como se hacía en el pgm. anterior. El control de si se trata de modificación o socio nuevo es realizado por la línea 2050 mediante las variables NS (número de socios actual) y NSN (número de socios en curso).

Dado que cada uno de los tres sistemas de video existen se identifica con un carácter, debe tenerse en cuenta a efectos del INPUT de la línea 2100.

Búsqueda de socios

Esta rutina permite informar sobre los datos de un socio y la situación de alquiler en que se encuentra. La rutina de búsqueda se ha implementado sobre la base de conocer su número de socio (número que coincide con el de registro) o su nombre (al menos parcialmente).

Con el fin de utilizar una rutina común a ambas opciones, se establece un bucle de búsqueda entre los límites L1 y L2 (3070).

Como la variable N$ (3030) recoge indistintamente el número de socio o su nombre, la línea 3040 a través de VAL identificar el tipo de información. Si VAL (N$) es cero se trata de una cadena, es decir un nombre y por tanto los límites de búsqueda se extienden a todo el fichero (L1 = 1 y L2 = NS), en caso contrario se trata de un número, es decir, solo habrá que buscar el registro que coincide con ese número y por ello la línea 3050 hace L1 = L2 = NS, de forma que el bucle solo de una vuelta.

Una vez localizado el socio se imprimen en pantalla sus datos y la situación de alquiler.

Para listar todos los socios basta con pulsar RETURN como nombre o número de socio.

Gestión de películas

La línea 020 obliga a identificar al socio por su número, esto no es problema puesto que se dispone de la rutina anterior para conocer dicho número en función del nombre.

Una vez leído el registro correspondiente por la línea 4050, se imprime su situación actual. La línea 4140 permite realizar tres operaciones, devolución de las películas, nuevo alquiler y salida con grabación de la nueva situación de alquiler y fecha, actualizada.

La devolución se realiza poniendo a "nada" las variables de campo correspondiente en las películas. El nuevo alquiler se realiza mediante la línea 4160, junto con 4210 y 4220 y por último la salida produce una grabación actualizada del socio.

Cierre y fin

Es análogo al utilizado en el programa anterior y por tanto, caben las mismas observaciones respecto al final, duplicado por 5070 y 670.

```
10 ´ ************** GESTION DE VIDEOCLUBS *********************
20 ´ **************   Fichero aleatorio   *********************
30 ´ **************         "VIDEO"       *********************
40 DATA ENERO,FEBRERO,MARZO,ABRIL,MAYO,JUNIO,JULIO,AGOSTO,SEPTIEMBRE,OCTUBRE,NOV
IEMBRE,DICIEMBRE
50 CLS$=CHR$(27)+"E"+CHR$(27)+"H"
60 F$="SOCIOS"
70 PRINT cls$ PRINT TAB(6)"***************** GESTION DE VIDEOCLUBS *************
********"
80 RESTORE DIM M$(12) FOR M=1 TO 12 READ M$(M) NEXT
90 s$=STRING$(61,"*") PRINT TAB(6)s$ PRINT
100 PRINT TAB(15)"Introducir fecha  DDMMAA   ", INPUT "",fecha$
110 IF LEN(fecha$)<>6 THEN 100
120 PRINT TAB(20), GOSUB 5130
130 PRINT PRINT PRINT TAB(20)"Fecha correcta  (S/N) ", INPUT z$
140 IF UPPER$(z$)<>"S" THEN 130
150 ´
500 ´                          M E N U
510
520 PRINT cls$ PRINT TAB(6)"***************** GESTION DE VIDEOCLUBS ************
*********"
530 PRINT TAB(6)S$
540 PRINT
550 PRINT TAB (30)"M  E  N  U"
560 PRINT TAB (30)"==========="
570 C=27 PRINT PRINT TAB(C)"1  ABRIR FICHERO"
580 PRINT PRINT TAB(C)"2  ALTA NUEVO SOCIO"
590 PRINT PRINT TAB(C)"3  BUSCAR SOCIO"
600 PRINT PRINT TAB(C)"4  GESTION DE PELICULAS"
610 PRINT PRINT TAB(C)"5  CIERRE DE LA SESION"
620 PRINT PRINT PRINT TAB(6)S$
630 PRINT PRINT TAB(28)"", INPUT "SELECCIONAR OPCION ",OP
```

```
640 ON OP GOSUB 1000,2000,3000,4000,5000
650 GOTO 500
660 PRINT CLS$
670 END
1000 '                                  ABRIR FICHERO
1010 '
1020 PRINT cls$
1030 OPEN "R",1,F$
1040 FIELD 1,25 AS NOM$,30 AS CL$,7 AS TEL$,1 AS SIS$,6 AS FE$,5 AS P$(1),5 AS P
$(2),5 AS P$(3),5 AS P$(4),5 AS P$(5)
1050 IF FIND$("NSOCIOS")="" THEN OPEN "O",2,"NSOCIOS" WRITE #2,NS CLOSE 2 GOTO 1
070
1060 OPEN "I",2,"NSOCIOS" INPUT #2,NS CLOSE 2
1070 CLOSE 1
1080 PRINT PRINT PRINT"FICHERO SOCIOS VERIFICADO  CONTIENE ",NS," FICHAS"
1090 PRINT GOTO 3220
1100 '
2000 '                          INTRODUCIR O MODIFICAR FICHA
2010 '
2020 PRINT CLS$ PRINT"ULTIMO SOCIO ",NS
2030 PRINT INPUT "NUMERO DE SOCIO ",NSN
2040 OPEN "R",1,F$
2050 IF NSN>NS THEN NS=NSN
2060 PRINT CLS$ PRINT"S O C I O  No ",NS
2070 INPUT "Nombre                ",N$
2080 INPUT "Domicilio             ",D$
2090 INPUT "Telefono              ",T$
2100 INPUT "Sistema  (V=V H S, B=BETA, D=2000)        ",SI$
2110 PRINT INPUT "datos correctos (S/N) ",Z$
2120 IF UPPER$(Z$)<>"S" THEN 2060
2130 LSET NOM$=N$
2140 LSET CL$=D$
2150 LSET FE$=FECHA$
2160 FOR K=1 TO 5 LSET P$(K)="" NEXT K
2170 LSET TEL$=T$
2180 LSET SIS$=SI$
2190 PUT 1,NS
2200 CLOSE
2210 RETURN
3000 '                                  BUSCAR FICHA
3010 PRINT CLS$,"BUSQUEDA DE SOCIO"
3020 PRINT"==================="
3030 INPUT "NOMBRE O NUMERO DE SOCIO ",N$
3040 IF VAL (N$)=0 THEN L1=1 L2=NS GOTO 3060
3050 IF VAL(N$)<=NS THEN L1=VAL(N$) L2=VAL(N$) ELSE PRINT CHR$(7) GOTO 3010
3060 OPEN "R",1,F$
3070 FOR K1=L1 TO L2
3080 GET 1,K1
3090 IF L1=L2 THEN 3110
3100 IF INSTR(NOM$,N$)=0 THEN 3190
3110 PRINT CLS$.PRINT"Socio No ",K1
3120 PRINT"Nombre          ",NOM$
3130 PRINT"Domicilio       ",CL$
3140 PRINT"Telefono        ",TEL$
3150 PRINT"Peliculas alquiladas el ", GOSUB 5100
3160 FOR k=1 TO 5 PRINT"No  ",P$(k) NEXT k
3170 PRINT PRINT PRINT PRINT
3180 GOTO 3240
3190 NEXT K1
3200 PRINT PRINT"NO HAY MAS FICHAS"
3210 CLOSE
3220 INPUT "PRESIONE <RETURN> PARA VOLVER AL MENU",Z$
3230 RETURN
3240 INPUT "INTRODUCIR   C=Continuar S=Salir ",Z$
3250 IF UPPER$(Z$)="C" THEN 3190
3260 IF UPPER$(Z$)="S" THEN 3210
4000 '                          GESTION PELICULAS
4010 PRINT CLS$
4020 INPUT  "NUMERO DE SOCIO ",N
4030 IF N>NS THEN PRINT"NO EXISTE, EL ULTIMO ES EL",NS GOTO 4190
4040 OPEN "R",1,F$
4050 GET 1,N
4060 PRINT"D ",NOM$,
4070 PRINT"C/",CL$
```

```
4080 PRINT PRINT"Tel ",TEL$,
4090 IF SIS$="V" THEN PRINT"Sistema  V H S" GOTO 4120
4100 IF SIS$="B" THEN PRINT"Sistema  BETA" GOTO 4120
4110 PRIN"Sistema  2000"
4120 PRINT PRINT"PELICULAS ALQUILADAS EL  ", GOSUB 5100
4130 FOR K=1 TO 5 PRINT"Pelicula No ",k," - ",P$(K) NEXT K
4140 PRINT INPUT "D=Devolucion, A=Alquiler, S=Salir ",Z$
4150 IF UPPER$(Z$)="D" THEN FOR K=1 TO 5 LSET P$(K)="" NEXT K PRINT"PELICULAS CA
NCELADAS"·GOTO 4140
4160 IF UPPER$(Z$)="A" THEN 4210
4170 LSET FE$=FECHA$ PUT 1,N
4180 CLOSE
4190 PRINT PRINT  INPUT "PRESIONAR <RETURN> PARA VOLVER AL MENU",Z$
4200 RETURN
4210 INPUT "NUMERO DE PELICULAS A ALQUILAR ",NP
4220 FOR K=1 TO NP PRINT "PELICULA ",K, INPUT "No  ",PEL$
4230 LSET P$(K)=PEL$
4240 NEXT K
4250 GOTO 4140
5000 '                    CERRAR FICHERO Y FIN PROGRAMA
5010 PRINT CLS$
5020 OPEN "O",2,"NSOCIOS" WRITE #2,NS CLOSE 2
5030 CLOSE
5040 PRINT"F I C H E R O S    C E R R A D O S"
5050 PRINT PRINT"NUMERO DE FICHAS ALMACENADAS",NS
5060 PRINT PRINT"F I N   D E    P R O G R A M A"
5070 END
5100 '                     Rutinas de conversion de fechas
5110 PRINT VAL(LEFT$(FE$,2))," / ",M$(VAL(MID$(FE$,3,2))," /",1900+VAL(RIGHT$(F
E$,2))
5120 RETURN
5130 PRINT VAL(LEFT$(FECHA$,2))," / ",M$(VAL(MID$(FECHA$,3,2))) " /",1900+VAL(RI
GHT$(FECHA$,2))
5140 RETURN
```

CAPITULO 7

FICHEROS DE ACCESO POR CLAVES

1. **INTRODUCCION**
2. **JETSAM**
3. **MANEJO DE LOS ARCHIVOS**
4. **CREACION DE FICHEROS INDEXADOS**
5. **APERTURA Y CIERRE**
6. **ESTRUCTURA Y ASIGNACION DE CAMPOS**
7. **FUNCIONES JETSAM**
8. **GRABACION Y LECTURA DE REGISTRO**
9. **EJEMPLO PRACTICO 1**
   Base de datos. Proveedores
10. **EJEMPLO PRACTICO 2**
    Gestión de biblioteca

7. Ficheros de acceso por claves

1. INTRODUCCION

Tras la lectura de los dos capítulos anteriores ha podido ver la gran diferencia entre los ficheros secuenciales y los de acceso aleatorio o directo. En estos últimos, cada registro es identificado por un número, de forma que si se asigna por ejemplo a cada cliente un número igual al de registro, la búsqueda es inmediata. En ocasiones y siguiendo con el ejemplo, resulta que para cada cliente existe más de un registro con lo que la búsqueda ha de hacerse mediante una expresión más o menos complicada en función de la disposición de datos en el archivo. Pero si la búsqueda se realiza por cualquier concepto distinto del número de registro, resulta que es necesario realizar una de tipo secuencial, puede ver un ejemplo en los dos programas anteriores, donde la búsqueda se realiza en esta forma, desaprovechándose en estos casos el acceso directo propiamente dicho. Como resumen podemos indicar que los ficheros aleatorios, son de acceso directo únicamente respecto del número de registros y no del contenido de cualquier campo, que habrá de realizarse secuencialmente.

Este problema de acceso ha dado origen a lo que se conoce como ficheros indexados o de acceso por claves. Estos ficheros poseen un índice al igual que un libro, que permiten "mirar" en dicho índice y acceder luego directamente a la información buscada. Se componen pues de dos ficheros uno de *índices* y otro de *datos*.

Ficheros de acceso por claves (indexados) { Fichero de índices / Fichero de datos

A medida que se introducen los datos propiamente dichos, con destino al fichero *datos*, el programa va creando un índice de registros con destino al fichero *índices*, de esta forma para localizar posteriormente un determinado registro, bastará con introducir el campo clave, el programa leerá el índice y cuando localice la clave introducida, accederá directamente al dato.

Aparentemente este método puede parecer largo y como consecuencia lento, pero no es así, tenga en cuenta que el fichero de índices es más corto que el de datos y por tanto se localiza antes una clave y su registro correspondiente, que por el método secuencial visto anteriormente.

2. JETSAM

El Mallard BASIC de los PCW gestiona los ficheros indexados a través de un módulo denominado JETSAM. También los usuarios de los CPC 6128 tienen acceso a estos ficheros a través de la instalación descrita en el capítulo 0, de forma que emulan en todo momento el trabajo de los PCW.

JETSAM tiene una limitación de hasta ocho tipos de clave, que corresponden a campos de un registro, cuya longitud no excederá de 32 bytes (caracteres) por clave.

El programa se encarga de seleccionar el o los campos que serán claves, su longitud y además el orden de prioridad de dichas claves (si hay varias). Esta disposición de claves ordenadas según una prioridad (Rango de clave), es necesaria puesto que si se realiza una búsqueda por una clave y el dato está repetido, será necesario buscar por una segunda clave e incluso una tercera, etc., hasta determinar el dato exacto. Este caso es muy típico en la búsqueda de nombres; la primera clave corresponde con el primer apellido, si está repetido se busca por la segunda, que corresponde con el segundo, si también existen otras personas con apellidos iguales, será necesario buscarla por una tercera, que correspondería al nombre, etc.

3. MANEJO DE LOS ARCHIVOS

Ya se han podido ver las ventajas de estos archivos respecto de los directos, los inconvenientes no son muchos y sin duda el más importante sobre todo para el principiante, es que poseen un cierto grado de complejidad en su estructuración y manejo, unido a un mayor número de comandos, que obliga a no perder de vista el manual durante la programación.

Hay una cuestión que debe ser tenida en cuenta en el trabajo con este tipo de ficheros y es que el concepto de número de registro no existe ya para el programador; sólo existe para JETSAM que los manipula automáticamente a través de un *puntero*.

El programa ordena al módulo que busque un registro con una determinada clave, cuando JETSAM la localiza, dirige hacia él su puntero, mientras dicho puntero permanezca apuntando, el registro podrá ser leído y/o modificado.

El manejo de ficheros de acceso por claves o indexados, se realiza generalmente a través de *funciones* en vez de comandos u órdenes, este es otro rasgo diferencial, respecto a los aleatorios. Las funciones dan como resultados valores y no acciones como las órdenes y como todas las funciones, han de ir acompañadas de una orden BASIC, generalmente la de asignación LET o bien de decisión IF.

```
LET var% = función (parámetros)
IF función (parámetros) condición THEN acción
```

Los valores generados por estas funciones y sus significados son los siguientes.

Código generado	*Significado*	
0	Operación correcta	
101 102 103	Se ha sobrepasado el final de claves Se ha sobrepasado el final de rangos Se ha sobrepasado la última clave	
105 115	No se ha encontrado la clave No está definida la posición actual	
130 131 132 133	El fichero está bloqueado por otro usuario No puede realizarse el bloqueo de escritura (clave) No puede realizarse el bloqueo de lectura No puede realizarse el bloqueo de escritura	

Resulta muy útil la inclusión en los programas que manejan ficheros indexados, de una rutina de comprobación de resultados como la indicada a continuación, y a la que debe llamarse cada vez que se utilice una de las funciones, de forma que se produzca una información sobre el éxito o fracaso de la operación ejecutada. En dicha rutina la variable RESUL habrá obtenido previamente su valor en el programa mediante alguna de las funciones JETSAM que serán estudiadas más adelante.

```
9000 ' RUTINA DE COMPROBACION DE RESULTADOS DE FUNCIONES JETSAM
9010 '
9020 DATA 0,OPERACION CORRECTA,101,Se ha sobrepasado el final de claves
9030 DATA 102,Se ha sobrepasado el final de rangos
9040 DATA 103,Se ha sobrepasado la ultima clave
9050 DATA 105,No se ha encontrado la clave
9060 DATA 115,No esta definida la posicion actual
9070 DATA 130, El fichero esta bloqueado por otro usuario
9080 DATA 133, No puede realizarse el bloqueo de escritura (CLAVE)
9090 DATA 132, No puede realizarse el bloqueo de lectura
9100 DATA 133, No puede realizarse el bloqueo de escritura
9110 RESTORE 9020  FOR k=1 TO 10
9120 READ COGEN,TEXTO$
9130 IF RESUL=COGEN THEN PRINT TEXTO$ GOTO 9150
9140 NEXT K
9150 RETURN
```

4. CREACION DE FICHEROS INDEXADOS

Tal como se vió en los capítulos anteriores los ficheros secuenciales y de acceso directo se crean con la orden OPEN si previamente no existían, en los indexados por el contrario antes de abrir un fichero, este ha debido ser creado, además es necesario antes de comenzar a trabajar el reservar un espacio de memoria suficiente para que JETSAM pueda realizar el mantenimiento de índices, ya que este se realiza en la propia memoria y solo es pasado al disco cuando se produce un cierre, o bien con una orden específica para tal fin.

La reserva de memoria mencionada se realiza en bloques de 128 bytes denominadas tampones (buffers), que en condiciones normales bastará con 5 ó 6. Debe tenerse en cuenta que una reserva muy elevada para estos buffers, supone una merma en la capacidad total del programa, no obstante una K equivale a 8 buffers.

La orden de reserva de tampones es precisamente BUFFERS seguida del número de ellos que quiera reservarse.

BUFFERS nº de tampones

Esta orden debe utilizarse al comienzo del programa, antes de realizar operaciones con ficheros.

Para la creación de un fichero indexado, se utiliza la orden CREATE, cuya forma general es:

CREATE nº fichero, "nombre fichero datos", "nombre fichero claves", 2, var. de longitud del fichero.

— nº fichero: Tiene el mismo significado que en los anteriores, identificar cada fichero con un número.

— "nombre fichero datos": Nombre que recibirá el fichero destinado a contener los datos.

— "nombre fichero claves": id. del de claves.

— 2: Número obligatorio (!).

— var. long. registro: Es una variable numérica que contiene la longitud de los registros y que debe definirse previamente.

Por supuesto los dos nombres entre comillas pueden ser sustituidos por variables alfanuméricas y la variable de longitud por un número.

La orden CREATE puede incluirse en un programa o ser utilizada como orden directa, los efectos son idénticos. Si se opta por incluirla en un programa solo debe usarse una vez, para ello será necesario controlar la existencia de los ficheros antes de ser creados, si no existen pasar a su creación pero en caso de existir la orden debe omitirse. Para ello debe utilizarse la orden FIND$ descrita en los capítulos anteriores. En caso de utilizarse como orden directa debe tenerse en cuenta que el fichero queda abierto, por lo que si no va a ser utilizado inmediatamente, debe procederse a su cierre.

La longitud de los registros se establece en la misma forma que en los aleatorios, utilizando la orden FIELD que define la longitud de cada campo, pero la longitud de un registro es ahora igual a la suma de la de sus campos más dos, debido a que

$$\text{Long.registro} = \Sigma \text{ long.campos} + 2$$

JETSAM reserva 2 bytes de cada registro para su uso, realizando internamente la interrelación registros-claves.

5. APERTURA Y CIERRE

La apertura de un fichero indexado se realiza con la orden básica OPEN, como en los anteriores, pero en este caso va seguida del indicador "K" (clave ≡ key), al igual que en los otros tipos de fichero se debía indicar "R" (aleatorios) y "O", "I" (secuenciales).

La orden completa es:

OPEN "K", nº fichero, "nomfich.datos", "nomfich.claves", 2, var. longitud

como ve su forma se ajusta a la orden de creación, siendo válidas las observaciones comentadas anteriormente.

El número de fichero no tiene porqué coincidir con el utilizado durante la creación, la orden OPEN es la que determina la identificación del fichero y no la de creación, es decir cualquier referencia posterior al fichero llevará el número indicado en OPEN.

El cierre se realizará con la orden CLOSE pero obligatoriamente seguida del número de fichero, por tanto no existe la orden CLOSE para cierre de todos los ficheros.

Dado que la actuación de los índices se va realizando en memoria y no son grabados en el fichero de claves hasta que el fichero de datos es cerrado con el CLOSE correspondiente, cualquier incidencia fortuita, como el olvido de dicho cierre, un fallo en el suministro de energía, etc., puede dar al traste con el fichero de claves y dejar como consecuencia "inútil" el de datos o dicho de otra forma más acertada, el fichero de datos será inconsistente (no correspondencia claves-datos).

Para paliar la situación descrita que lejos de ser poco probable y por las leyes de Murphy será frecuente y en el momento más inoportuno, se dispone de la orden CONSOLIDATE que permite grabar el contenido de los buffers de claves de memoria, en el fichero de claves del disco. Esta orden no implica un cierre del fichero, solo la actualización en disco descrita, es decir equivale a un CLOSE seguido de un OPEN.

6. ESTRUCTURA Y ASIGNACION DE CAMPOS

Se realiza de la misma forma que en los aleatorios, los campos se definen con FIELD ignorando totalmente los 2 bytes reservados por JETSAM. La asignación se realiza con LSET, RSET y MID$ como se explicó en el capítulo anterior.

7. FUNCIONES JETSAM

Todo lo visto hasta ahora coincide sensiblemente con los ficheros de acceso directo, son las funciones JETSAM lo que constituye su gran diferencia en el manejo de los ficheros indexados, a continuación se describen dichas funciones.

— Funciones generales

— ADDREC. Sirve para escribir los datos almacenados en las variables de campo (definidas en FIELD) en el siguiente registro al último que exista en el fichero. El puntero queda señalando a la nueva posición y se añade la clave del nuevo registro en el fichero de claves (temporalmente en memoria). Su forma general es

Var. = ADDREC (nº fichero, 2, rango, clave)

Dado que ADDREC es una función se ha utilizado la asignación LET (implícita) para introducir el código generado en la variable numérica, que denominamos genéricamente VAR.

El valor incluido en "clave" (campo), es situado de acuerdo con el "rango" indicado, actualizándose dicho rango de claves.

Si el fichero ha de estar clasificado por varios claves, detrás de la instrucción descrita debe utilizarse la siguiente.

var. = ADDKEY (nº fichero, 0, rango, clave, nº registro)

Su forma es similar a la anterior, pero para poder determinar el número de registro que utilizó ADDREC, ya que debe incorporarse una clave nueva en el registro, será necesario utilizar la función

var. = FETCHREC (nº fichero)

la variable indicada en la sentencia tomará el número de registro al que señala el puntero. Dicha variable y su contenido podrá ser usado en la función anterior y en otras.

Debe tenerse en cuenta que ADDREC añade nuevos registros pero no permite modificar registros ya existentes (intermedios), para ello se utilizan las órdenes GET y PUT para leer y escribir respectivamente utilizando la forma general.

GET nº fichero
PUT nº fichero

Evidentemente, antes de utilizar PUT se habrán cargado las variables de campo con los datos correspondientes, tal como se hacía en los ficheros aleatorios.

Al igual que la función FETCHREC nos indicaba el número de registro al que apuntaba el puntero, existe otro similar que permite conocer el rango de la última clave introducida, su forma general es

var. = FETCHRANK (nº de fichero)

Así como FETCHRANK nos devuelve el rango, FETCHKEY$ devuelve el contenido de la última clave, su forma general es

var $ = FETCHKEY$ (nº de fichero)

El borrado de claves es siempre una operación delicada. Si un registro solo dispone de una clave y esta es borrada, implica que dicho registro queda eliminado, no físicamente, pero sí prácticamente puesto que la no existencia de su clave imposibilita definitivamente su acceso.

En el caso de cambiar una clave siempre es preferible previamente crear la nueva, para posteriormente borrar la vieja. La adición de la nueva se realizará mediante la función ADDKEY descrita anteriormente y el borrado de la vieja mediante

var. = DELKEY (nº fichero, 0, posición)

donde el parámetro *posición* es optativo, de forma que si no se indica se borrará la que señale el puntero, por ello previamente habrá de situarse el puntero conveniente, mediante las funciones de búsqueda, descritas en el siguiente epígrafe.

La utilización de claves repetidas es optativa, si se desea impedir la repetición debe utilizarse la función

var. = RANKSPEC (nº fichero, nº clave, 0)

— Funciones de búsqueda

Tanto en los ficheros secuenciales, como aleatorios, todo lo relativo a la búsqueda de datos pasa forzosamente por el concepto de número de registro. En los secuenciales se busca desde el principio, uno tras otro, el registro que cumpla las condiciones indicadas, en los aleatorios se puede proceder de forma análoga, o bien acceder a un registro determinado, conociendo su número.

En los ficheros indexados, el concepto de número de registro como se ha dicho, no es necesario, la búsqueda se realiza a través de las claves, de forma indirecta, JETSAM se encarga de encontrar el registro que obedece a las claves indicadas, para ello se dispone de varias funciones, que permiten conocer la existencia de registros con la misma clave, claves distintas dentro de un rango o bien la no existencia de la clave buscada, pasemos a una descripción somera de las mismas.

SEEKKEY permite buscar en el fichero el "primer" registro cuya clave se corresponda con la indicada, dentro del rango dado. Debe tenerse en cuenta que existe un solo fichero de claves, a partir de la última clave de un rango, viene la primera clave del siguiente rango y así sucesivamente.

La búsqueda en un rango puede dar positiva en él, o en otro rango, siendo el resultado evidentemente distinto, será interesante conocer no sólo si se ha encontrado, sino también donde.

La forma general de la función es

var. SEEKKEY (nº fichero, 0, rango, clave)

Un problema común dentro de un fichero es la existencia de varios registros, con la misma clave (nombres repetidos, etc), dado que SEEKKEY busca el primero, si este no es el deseado o bien es preciso conocer todos los que correspondan a esa clave, debe utilizarse SEEKNEXT que recorre todo el fichero buscando todos los registros que correspondan a la clave y rango dados.

Su forma general es

var. = SEEKNEXT (nº fichero, 0, origen búsqueda)

El origen de búsqueda permite realizar la misma desplazando el puntero al lugar indicado. Este origen es opcional, de forma que si se omite, la búsqueda se realizará a partir de la posición a la que en ese momento señale el puntero.

Existe otra función idéntica a la anterior y con la misma sintaxis, la diferencia es que la búsqueda la realiza por la clave anterior a la indicada.

Su forma general es

var. = SEKKPREV (nº fichero, 0, origen búsqueda)

Una última función, útil en el caso de claves repetidas, es SEEKSET, esta función va saltándose las claves que se repiten, hasta encontrar una nueva, su síntaxis es

var. = SEEKSET (nº fichero, 0, origen de búsqueda)

como puede verse su sintaxis es idéntica a SEEKNEXT.

8. GRABACION Y LECTURA DE REGISTROS

Las órdenes de grabación y lectura en los ficheros indexados son las mismas que en los aleatorios, PUT y GET respectivamente. La forma de utilizarlas, asignación de variables de campo, etc., es también idéntica, lo único que varía es que aquí no se manejen números de registros sino que ambas operaciones se realizan sobre el registro al que apunta el puntero en ese momento y por tanto su síntaxis es simplemente

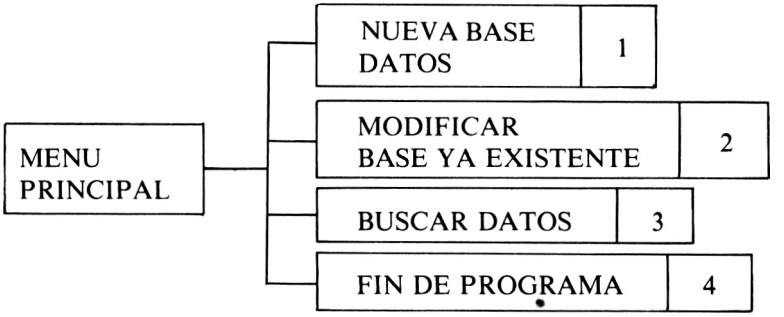
GET nº fichero (leer)
PUT nº fichero (grabar)

Sin duda la mera descripción de las sentencias y funciones, que permiten trabajar con ficheros indexados, no es suficiente para permitir su uso inmediato. Un ejemplo práctico permite aclarar los conceptos y fijar ideas, a tal fin en los siguientes epígrafes se han incluido dos programas completos. El primero constituye una base de datos para proveedores y el segundo permite la gestión de una biblioteca, programa que ya fue desarrollado anteriormente con ficheros secuenciales y que permite ahora su estudio comparativo.

9. EJEMPLO PRACTICO 1. Base de datos. Proveedores

Al igual que en los capítulos precedentes y a modo de colofón se incluye un ejemplo de utilización de ficheros indexados. Se trata de un fichero de proveedores que recoge los datos habituales, de nombre, dirección, producto suministrado y precio actual. El programa permite la búsqueda por nombre, producto y precio, este último definido por un entorno (máximo y mínimo). El programa aún siendo específico en el sentido expuesto, resulta de fácil adaptación a cualquier otro caso, como los estudiados en capítulo anteriores, agenda, biblioteca, etc., su mayor potencia se da sin duda dentro de los programas de gestión, para los cuales los ficheros indexados resultan los más apropiados.

La estructura básica del programa consiste en un Menú director (menú principal) del que parten cuatro rutinas principales.



La primera de las rutinas principales permite crear una o varias bases de datos (ficheros) distintas, obviamente cada uno de estos ficheros de datos, irá acompañado de su correspondiente fichero de índices, como verá en la rutina 3000, el fichero consta de 6 caracteres como máximo y el de indice, tiene el mismo nombre anteponiendo "IN", por ejemplo

Fichero de datos "COMPRA"
Fichero de indice "INCOMPRA"

de esta forma, en caso de utilizar varios ficheros de datos, es fácil identificar sus correspondientes de índices.

Todas las rutinas y subrutinas van señaladas con comentarios a fin de facilitar el seguimiento del programa.

Si no desea teclear el listado, dispone de un disco que contiene este y otros programas desarrollados en el libro.

```
10 ' ************************************************************
20 ' *************   BASE DE DATOS. PROVEEDORES   ******************
30 ' *************  Fichero de acceso por claves ******************
40 ' *************              "PROVE"           ******************
50 ' ************************************************************
60 BUFFERS 6
70 CLS$=CHR$(27)+"E"+CHR$(27)+"H"
80 DEF FN LOC$(F,C)=CHR$(27)+"Y"+CHR$(32+F)+CHR$(32+C)
90 ' M E N U     PRINCIPAL
100 '
110 PRINT CLS$
120 PRINT FN LOC$(5,30);"M E N U"
130 PRINT FN LOC$(6,29);"=========="
140 PRINT FN LOC$(8,18);"1. CREAR UNA NUEVA BASE DE DATOS"
150 PRINT FN LOC$(10,18);"2. MODIFICAR BASE DE DATOS CREADA"
160 PRINT FN LOC$(12,18);"3. BUSCAR DATOS"
170 PRINT FN LOC$(14,18);"4. FIN DE PROGRAMA"
180 PRINT FN LOC$(21,18);"Seleccionar opcion ";:INPUT OP
190 IF OP<1 OR OP>4 THEN 90
200 IF OP=4 THEN 300
210 ON OP GOSUB 1000,1270,4000
220 GOTO 90
300 ' ............................... FIN DE PROGRAMA
310 PRINT CLS$
320 PRINT FN LOC$(12,25);"F  I  N"
330 '  .
340 CLOSE 1
350 END
400 ' ........................... IMPRIMIR REGISTRO
410 '
420 PRINT CLS$
430 PRINT "Nombre ......... ";NOM$
440 PRINT "Direccion ...... ";DIR1$
450 PRINT "            ...... ";DIR2$
460 PRINT "            ...... ";DIR3$
470 PRINT "Telefono  ...... ";TEL$
480 PRINT "Producto  ...... ";PRO$
490 PRINT "Precio  ........ ";PRE$
500 PRINT:PRINT:PRINT:PRINT"Presione una tecla para continuar"
510 IF INKEY$="" THEN 510
520 RETURN
1000 ' ..... ....... ...... CREACION DE LA BASE DE DATOS
1010 '
1020 PRINT CLS$:INPUT "NOMBRE DE LA BASE DE DATOS (Max. 6 caracteres)"
,FD$
1030 IF LEN(FD$)>6 THEN 1020
1040 FD$=UPPER$(FD$):FI$="IN"+FD$
1050 CREATE 1,FD$,FI$,2
1060 FIELD 1,25 AS NOM$,25 AS DIR1$,15 AS DIR2$,15 AS DIR3$,15 AS TEL$
,20 AS PRO$,6 AS PRE$
1070 PRINT:PRINT"FICHEROS CREADOS"
1080 PRINT"FICHERO DE DATOS  = ";FD$
1090 PRINT"FICHERO DE INDICES= ";FI$
1100 PRINT:PRINT:INPUT "INTRODUCIR DATOS (S/N) ";Z$
1110 IF UPPER$(Z$)="S" THEN 1140
1120 CLOSE 1
1130 RETURN
1140 ' ..................................... ENTRADA DE DATOS
1150 GOSUB 2280
1160 IF UPPER$(XNOM$)="FIN" THEN 1120 ELSE 1150
1170 ' ..................................... ABRIR FICHEROS
1180 '
1190 PRINT CLS$
1200 INPUT "Nom. fichero datos (RETURN si es conocido por el pgm.) ";F
F$
```

```
1210 IF LEN(FF$)<7 AND FF$<>"" THEN FD$=FF$
1220 IF FIND$(FD$)="" THEN PRINT"NO EXISTE ESE FICHERO ";FD$:GOTO 1200
1230 FI$="IN"+FD$
1240 OPEN "K",1,FD$,FI$,2
1250 FIELD 1,25 AS NOM$,25 AS DIR1$,15 AS DIR2$,15 AS DIR3$,15 AS TEL$
,20 AS PRO$,6 AS PRE$
1260 RETURN
1270 ' ................................ MANEJO DE REGISTROS
1280 '
1290 GOSUB 1170
1300 GOSUB 2000
1310 IF OP1=5 THEN 1340
1320 ON OP1 GOSUB 2110,2280,2620,2510
1330 GOTO 1300
1340 CLOSE 1
1350 RETURN
2000 ' ....................... MENU DE MANEJO DE REGISTROS
2010 '
2020 PRINT CLS$
2030 PRINT FN LOC$(8,18);"1. VER REGISTROS"
2040 PRINT FN LOC$(10,18);"2. AGREGAR REGISTROS"
2050 PRINT FN LOC$(12,18);"3. MODIFICAR REGISTROS"
2060 PRINT FN LOC$(14,18);"4. BORRAR REGISTRO"
2070 PRINT FN LOC$(16,18);"5. VOLVER AL MENU PRINCIPAL"
2080 PRINT FN LOC$(21,18);"Seleccionar opcion ";:INPUT OP1
2090 IF OP1<1 OR OP1>5 THEN 2080
2100 RETURN
2110 ' Examen de registros
2120 PRINT cls$
2130 PRINT FN LOC$(8,18);"1. NOMBRES (orden alfabetico)"
2140 PRINT FN LOC$(10,18);"2. PRODUCTOS (orden alfabetico)"
2150 PRINT FN LOC$(12,18);"3. PRECIOS (de menor a mayor)"
2160 PRINT FN LOC$(21,18);"Seleccionar opcion ";:INPUT OP2
2170 IF OP2<1 OR OP1>3 THEN 2160
2180 RES=SEEKRANK(1,0,OP2)
2190 IF RES<>0 THEN GOTO 2270
2200 GET 1
2210 GOSUB 400
2220 RES =SEEKNEXT (1,0)
2230 IF RES=102 OR RES=103 THEN 2270
2240 GET 1
2250 GOSUB 400
2260 GOTO 2220
2270 RETURN
2280 ' ....... Carga de datos y asignacion variables de campo
2290 '
2300 PRINT CLS$:PRINT"Introducir FIN para terminar"
2310 INPUT "Nombre ..... ... ";XNOM$
2320 IF UPPER$(XNOM$)="FIN" THEN 2500
2330 LSET NOM$=XNOM$
2340 INPUT "Direccion ...... ";XDIR1$
2350 LSET DIR1$=XDIR1$
2360 INPUT "            ..... ";XDIR2$
2370 LSET DIR2$=XDIR2$
2380 INPUT "           ...... ";XDIR3$
2390 LSET DIR3$=XDIR3$
2400 INPUT "Telefono  ...... ";XTEL$
2410 LSET TEL$=XTEL$
2420 INPUT "Producto  ...... ";XPRO$
2430 LSET PRO$=XPRO$
2440 INPUT "Precio  ........ ";XPRE$
2450 LSET PRE$=XPRE$
2460 RES=ADDREC(1,2,1,NOM$)
2470 rango=FETCHREC(1)
2480 RES=ADDKEY(1,0,2,PRO$,rango)
```

```
2490 RES=ADDKEY(1,0,3,PRE$,rango)
2500 RETURN
2510 ' Borrar registro
2520 PRINT cls$
2530 GOSUB 2960
2540 IF OP3=4 THEN 2610
2550 R=FETCHREC(1)
2560 RES=DELKEY(1,0,1,NOM$,R)
2570 RES=DELKEY(1,0,2,PRO$,R)
2580 RES=DELKEY(1,0,3,PRE$,R)
2590 PRINT.PRINT"Registro BORRADO !, presionar una tecla para continua
r"
2600 IF INKEY$="" THEN 2600
2610 RETURN
2620 ' Modificacion de registros
2630 GOSUB 2960
2640 IF OP3=4 OR CON=1 THEN 2950
2650 RNOM$=NOM$
2660 RPRE$=PRE$
2670 RPRO$=PRO$
2680 PRINT CLS$.PRINT"Introducir datos a modificar o RETURN para no al
terar"
2690 PRINT"Nombre actual ";NOM$
2700 INPUT "nuevo ";NNOM$
2710 IF NNOM$<>"" THEN LSET NOM$=NNOM$
2720 PRINT"Direccion actual ";DIR1$
2730 INPUT "nueva ";NDIR1$
2740 IF NDIR1$<>"" THEN LSET DIR1$=NDIR1$
2750 PRINT"Id. actual       ",DIR2$
2760 INPUT "nueva ",NDIR2$
2770 IF NDIR2$<>"" THEN LSET DIR2$=NDIR2$
2780 PRINT"Id. actual       ";DIR3$
2790 INPUT "nueva ";NDIR3$
2800 IF NDIR3$<>"" THEN LSET DIR3$=NDIR3$
2810 PRINT"Producto  actual ";PRO$
2820 INPUT "nuevo ";NPRO$
2830 IF NPRO$<>"" THEN LSET PRO$=NPRO$
2840 PRINT"Precio actual ";PRE$
2850 INPUT "nuevo ";NPRE$
2860 IF NPRE$<>"" THEN LSET PRE$=NPRE$
2870 R=FETCHREC(1)
2880 RES=DELKEY(1,0,1,RNOM$,R)
2890 RES=DELKEY(1,0,2,RPRO$,R)
2900 RES=DELKEY(1,0,3,PRE$,R)
2910 RES=ADDREC(1,2,1,NOM$)
2920 R=FETCHREC(1)
2930 RES=ADDKEY(1,0,2,PRO$,R)
2940 RES=ADDKEY(1,0,3,PRE$,R)
2950 RETURN
2960 ' ..................................... MENU DE BUSQUEDA
2970 '
2980 CON=0
2990 PRINT CLS$:CONTROL$=""
3000 PRINT FN LOC$(5,14);"M E N U   DE   B U S Q U E D A"
3010 PRINT FN LOC$(6,14);"==============================="
3020 PRINT FN LOC$(8,18);"1. BUSCAR POR NOMBRES"
3030 PRINT FN LOC$(10,18);"2. BUSCAR POR PRODUCTOS"
3040 PRINT FN LOC$(12,18);"3. BUSCAR POR PRECIOS"
3050 PRINT FN LOC$(14,18);"4. SALIR DE ESTE MENU"
3060 PRINT FN LOC$(21,18);"Seleccionar opcion ";:INPUT OP3
3070 IF OP3<1 OR OP3>4 THEN 3060
3080 IF UPPER$(CONTROL$)="S" OR OP3=4 THEN 3100
3090 ON OP3 GOSUB 3110,3260,3410
3100 RETURN
3110 ' Busqueda por nombres
```

```
3120 '
3130 INPUT "Introducir nombre ";BNOM$
3140 BNOM$=BNOM$+SPACE$(25-LEN(BNOM$))
3150 RES=SEEKKEY(1,0,1,BNOM$)
3160 IF RES<>0 THEN CON=1:PRINT"No existe ese nombre o es incorrecto":
GOTO 3230
3170 GET 1
3180 GOSUB 400
3190 GOSUB 3560
3200 IF UPPER$(CONTROL$)="S" THEN 3250
3210 RES=SEEKNEXT(1,0)
3220 GOTO 3160
3230 PRINT"PRESIONE UNA TECLA PARA CONTINUAR"
3240 IF INKEY$="" THEN 3240
3250 RETURN
3260 ' Busqueda por productos
3270 '
3280 INPUT "nombre del producto ";BPRO$
3290 BPRO$=BPRO$+SPACE$(20-LEN(BPRO$))
3300 RES=SEEKKEY(1,0,2,BPRO$)
3310 IF RES<>0 THEN CON=1:PRINT"No existe ese producto o es incorrecto
":GOTO 3380
3320 GET 1
3330 GOSUB 400
3340 GOSUB 3560
3350 IF UPPER$(CONTROL$)="S" THEN 3400
3360 RES=SEEKNEXT(1,0)
3370 GOTO 3310
3380 PRINT"PRESIONE UNA TECLA PARA CONTINUAR"
3390 IF INKEY$="" THEN 3390
3400 RETURN
3410 ' Busqueda por precios
3420 '
3430 INPUT "Limite superior de precios ";PS%
3440 INPUT "Limite inferior de precios ";PI%
3450 FOR P%=PI% TO PS%
3460 P$=STR$(P%):P$=RIGHT$(P$,LEN(P$)-1):P$=P$+SPACE$(6-LEN(P$))
3470 RES=SEEKKEY(1,0,3,P$)
3480 IF RES<>0 THEN PRINT FN LOC$(10,60);"E S P E R E",P%:GOTO 3540
3490 GET 1
3500 GOSUB 400:GOSUB 3560
3510 IF UPPER$(CONTROL$)="S" THEN P%=PS%:GOTO 3540
3520 RES=SEEKNEXT(1,0)
3530 GOTO 3480
3540 NEXT P%
3545 PRINT FN LOC$(25,1)
3550 RETURN
3560 ' Comprobacion de registros
3570 PRINT
3580 INPUT "REGISTRO CORRECTO (S/N) ";CONTROL$
3590 IF UPPER$(CONTROL$)<>"S" AND UPPER$(CONTROL$)<>"N" THEN 3570
3600 RETURN
4000 ' ............................ BUSQUEDA DE REGISTROS
4010 '
4020 GOSUB 1170
4030 GOSUB 2960
4040 IF OP=4 THEN 4080
4050 PRINT
4060 INPUT "Continuar la busqueda (S/N) ";Z$
4070 IF UPPER$(Z$)="S" THEN 4030
4080 CLOSE 1
4090 RETURN
4100 RETURN
```

10. EJEMPLO PRACTICO 2. Gestión de biblioteca (JETSAM)

En el capítulo cinco se desarrolló un programa de gestión de biblioteca, utilizando ficheros secuenciales no muy apropiados a este tipo de aplicación, por ello se incluye aquí una nueva versión de dicho programa, utilizando ficheros de acceso por claves (indexados).

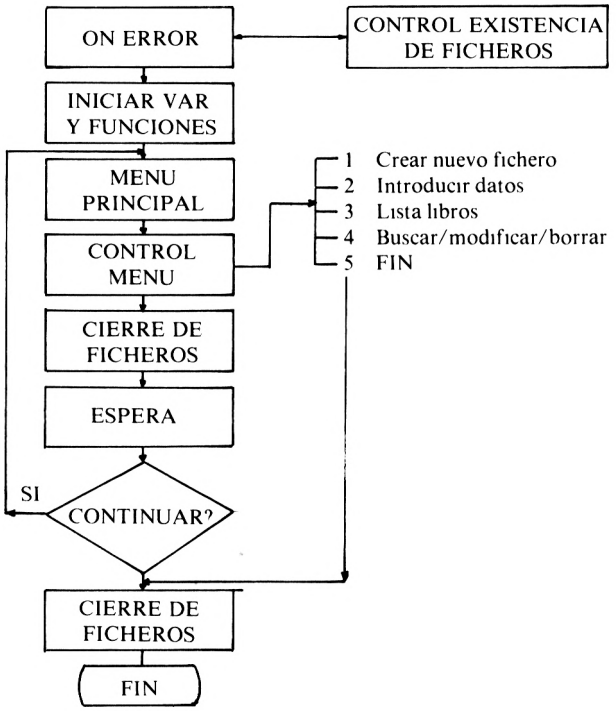
Se ha optado por la forma más simple de presentación de textos, dejando al lector la posibilidad de su mejora. Se ha tratado en todo momento de acortar en lo posible la longitud de los programas con el fin de facilitar su paso al ordenador.

Como verá el programa se ha estructurado como una colección de subrutinas, direccionadas desde un corto programa principal (300-450) precedido de una inicialización de variables y funciones, utilizadas por todo él (80-200).

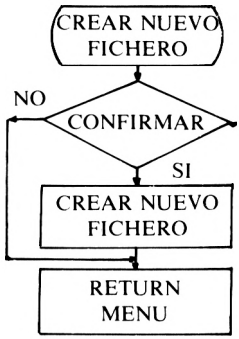
Cada subrutina está convenientemente etiquetada de forma que resulta fácil su estudio por separado.

Con el fin de facilitar el seguimiento del flujo del programa se incluyen a continuación los diagramas tanto del programa principal como de las cinco opciones básicas del Menú principal.

FLUJO PROGRAMA PRINCIPAL

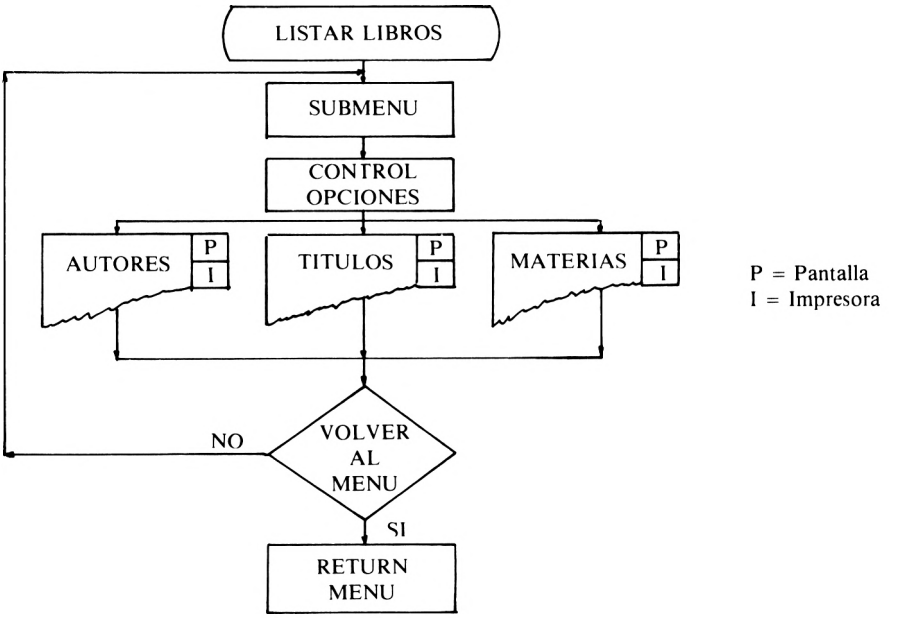


FLUJO OPCION 1 NUEVO FICHERO

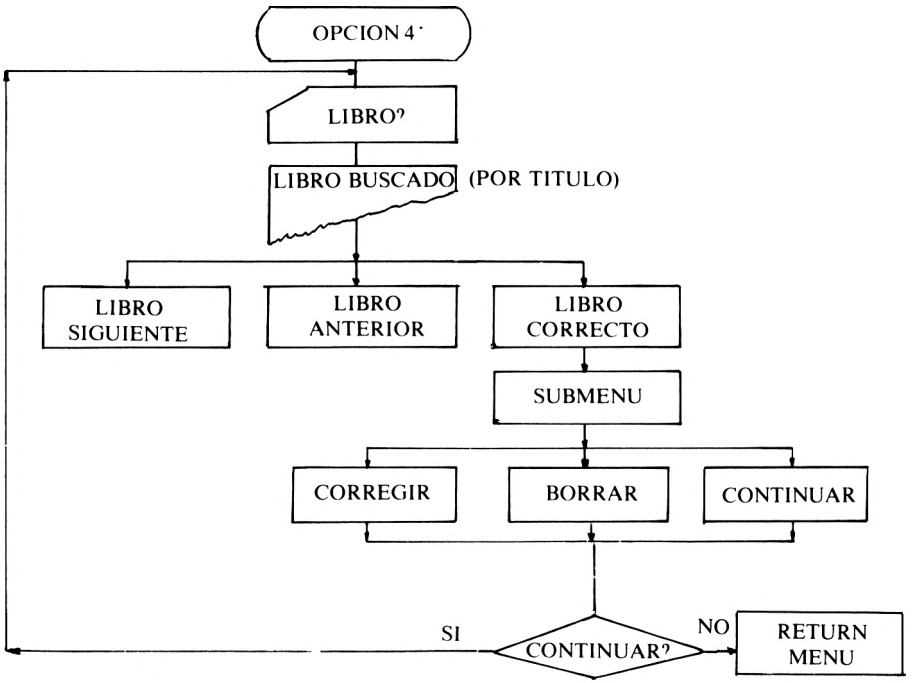


FLUJO OPCION 2 INTRODUCIR DATOS

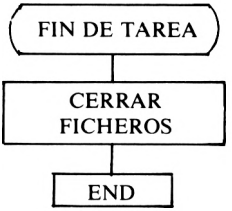
FLUJO OPCION 3 LISTADO DE LIBROS



FLUJO OPCION 4 BUSCAR/CORREGIR/BORRAR



FLUJO OPCION 5 FIN DE LA TAREA



```
10 ´ ***********************************************
20 ´ ******* GESTION DE BIBLIOTECA CON JETSAM *******
30 ´ ********  Fichero de acceso por claves   *******
40 ´ ********          "BIBLIOJS"             *******
50 ´ ***********************************************
60 ON ERROR GOTO 3110
100 ´.................................. INICIALIZACION
110 ´
120 CLS$=CHR$(27)+"E"+CHR$(27)+"H"
130 MASLINEA$=CHR$(27)+"v":MENOSLINEA$=CHR$(27)+"w"
140 IVIDEO$=CHR$(27)+"p":NVIDEO$=CHR$(27)+"q"
150 CONT$="PRESIONE UNA TECLA PARA CONTINUAR"
160 SINO$=IVIDEO$+" PRESIONE S o N (si/no) "+NVIDEO$
170 S$=STRING$(75,"-")
180 CONTINUA$="DESEA CONTINUAR"
190 IMPRE$="PRESIONAR <P> PARA IMPRESORA, CUALQUIER OTRA PAR
A PANTALLA"
200 SCROLL$="UTILZAR <ALT>-<S> PARA DETENER SCROLL PANTALLA"
210 ESPACIO$="PRESIONAR ESPACIO PARA VOLVER AL MENU"
220 f0%=0:f1%=1
230 F$="LIBROS.DAT":FI$="ILIBROS.DAT"
240 LI$="INDICE LIBRERIA"
250 BUFFERS 10:LONG%=92
260 PRINT MASLINEA$
300 ´  .    ....  ...  .. DIRECIONAMIENTO SUBRUTINAS
310 ´
320 GOSUB 500
330 GOSUB 610
340 ON OP% GOSUB 800,900,1200,2000,410
350 GOSUB 3050
360 PRINT CLS$
370 PRINT ESPACIO$
380 GOSUB 690
390 IF K$=" " THEN 320
400 ´
410 GOSUB 3050
420 PRINT MENOSLINEA$
430 END
500 ´    ....... . .. ....  ............ ..  M E N U
510 ´
520 PRINT cls$
530 PRINT TAB (35)IVIDEO$+"  M E N U  "+NVIDEO$:PRINT
540 PRINT TAB(30)"1 - CREAR NUEVO FICHERO":PRINT
550 PRINT TAB(30)"2 - Introducir datos":PRINT
560 PRINT TAB(30)"3 - Listar libros":PRINT
570 PRINT TAB(30)"4 - Buscar / Modificar / Borrar".PRINT
580 PRINT TAB(30)"5 - F   I   N":PRINT
590 MENU%=5
600 RETURN
610 ´ ......................  CONTROL ENTRADA MENU
620 ´
630 K$=""
640 WHILE K$="" OR VAL(K$)<1 OR VAL(K$)>MENU%
650 K$=INKEY$
```

```
660 WEND
670 OP%=VAL(K$)
680 RETURN
690 ' ............................ RUTINA DE ESPERA
700 '
710 K$="":WHILE K$="":K$=INKEY$:WEND
720 RETURN
800 ' ..................... CREACION DE NUEVO FICHERO
810 '
820 PRINT CLS$;"CREANDO NUEVO FICHERO"
830 PRINT :PRINT"PRESIONAR <N> PARA PARAR U OTRA TECLA PARA
CONTINUAR"
840 GOSUB 690
850 IF UPPER$(K$)="N" THEN 870 ELSE PRINT CLS$
860 CREATE F1%,F$,FI$,2,LONG%,LI$
870 PRINT CLS$
880 CLOSE F1%
890 RETURN
900 ' ........... RUTINA ENTRADA DE DATOS DE REGISTRO
910 '
920 PRINT CLS$
930 PRINT TAB(5)"I N T R O D U Z C A   L O S   D A T O S"
940 GOSUB 3000
950 PRINT
960 LINE INPUT"AUTOR (Apellidos, Nombre) ";A$
970 LINE INPUT"TITULO DEL LIBRO ......   ";T$
980 LINE INPUT"M A T E R I A  ........... ";M$
990 LSET AUTOR$=A$:LSET TITULO$=T$·LSET MAT$=M$
1000 IF F0%=1 THEN  RETURN
1010 ERR%=ADDREC (F1%,0,0,UPPER$(LEFT$(AUTOR$,6)))
1020 IF ERR%<>0 THEN GOSUB 1130
1030 RN%=FETCHREC(F1%)
1040 ERR%=ADDKEY(F1%,0,1,UPPER$(LEFT$(TITULO$,6)),RN%)
1050 IF ERR%<>0 THEN GOSUB 1130
1060 ERR%=ADDKEY(F1%,0,2,UPPER$(LEFT$(MAT$,6)),RN%)
1070 IF ERR%<>0 THEN GOSUB 1130
1080 PRINT
1090 PRINT CONTINUA$.PRINT SINO$
1100 GOSUB 690
1110 IF UPPER$(K$)="S" THEN 950
1120 RETURN
1130 ' ... ............ .     ...   CONTROL DE ERRORES
1140 '
1150 PRINT
1160 PRINT"SE HA DETECTADO EL ERROR CODIGO ",ERR%
1170 RETURN
1200 ' ......... .... .  ..   .     LISTADO DE LIBROS
1210 '
1220 PRINT CLS$+IVIDEO$+"   LISTADO  DE  LIBROS   "
1230 PRINT NVIDEO$
1240 GOSUB 3000
1250 PRINT "LISTAR POR·"
1260 PRINT TAB(9);"1 - AUTOR"
1270 PRINT TAB(9);"2 - TITULO"
```

```
1280 PRINT TAB(9);"3 - MATERIA"
1290 MENU%=3:PRINT
1300 GOSUB 610
1310 PRINT CLS$
1320 ON OP% GOSUB 1400,1600,1800
1330 PRINT"QUIERE VOLVER AL MENU?":PRINT SINO$
1340 GOSUB 690
1350 IF UPPER$(K$)="N" THEN PRINT CLS$:GOTO 1250
1360 OP%=0
1370 RETURN
1400 ' ........................ LISTADO POR AUTORES
1410 '
1420 PRINT CLS$
1430 PRINT IVIDEO$+"  LISTADO POR AUTORES  ":PRINT NVIDEO$
1440 PRINT CLS$;IMPRE$
1450 PRINT SCROLL$
1460 GOSUB 690
1470 IF UPPER$(K$)="P" THEN PR%=1 ELSE PR%=0
1480 PRINT CLS$
1490 ERR%=SEEKRANK(F1%,0,0)
1500 IF ERR%<>0 THEN GOSUB 1130
1510 WHILE ERR%=0 OR ERR%=101
1520 GET F1%
1530 GOSUB 4000
1540 ERR%=SEEKNEXT(F1%,0)
1550 WEND
1560 RETURN
1600 ' ................ ........ LISTADO POR TITULOS
1610 '
1620 PRINT CLS$
1630 PRINT IVIDEO$+"  LISTADO POR TITULOS  ":PRINT NVIDEO$
1640 PRINT IMPRE$
1650 PRINT SCROLL$
1660 GOSUB 690
1670 IF UPPER$(K$)="P" THEN PR%=1 ELSE PR%=0
1680 PRINT CLS$
1690 ERR%=SEEKRANK(F1%,0,1)
1700 IF ERR%<>0 THEN GOSUB 1130
1710 WHILE ERR%=0 OR ERR%=101
1720 GET F1%
1730 GOSUB 4100
1740 ERR%=SEEKNEXT(F1%,0)
1750 WEND
1760 RETURN
1800 ' .. .... .   ...   .      LISTADO POR MATERIAS
1810 '
1820 PRINT CLS$
1830 PRINT IVIDEO$+"  LISTADO POR MATERIAS ":PRINT NVIDEO$
1840 PRINT IMPRE$
1850 PRINT SCROLL$
1860 GOSUB 690
1870 IF UPPER$(K$)="P" THEN PR%=1 ELSE PR%=0
1880 PRINT CLS$
1890 ERR%=SEEKRANK(F1%,0,1)
```

```
1900 IF ERR%<>0 THEN GOSUB 1130
1910 WHILE ERR%=0 OR ERR%=101
1920 GET F1%
1930 GOSUB 4200
1940 ERR%=SEEKNEXT(F1%,0)
1950 WEND
1960 RETURN
2000 ' ............ .. ..... . .. BUSQUEDA DE LIBROS
2010 PRINT CLS$;TAB(25)"BUSCAR LIBROS"
2020 GOSUB 3000
2030 PRINT:PRINT "INTRODUZCA EL TITULO ";
2040 LINE INPUT T$:PRINT
2050 KEY$=UPPER$(LEFT$(T$+"      ",6))
2060 ERR%=SEEKKEY(F1%,0,1,KEY$)
2070 IF ERR%<>0 THEN GOSUB 3190:GOTO 2180
2080 GET F1%
2090 PRINT CLS$;:GOSUB 4000
2100 GOSUB 2430
2110 PRINT
2120 PRINT "Que desea realizar?"
2130 PRINT TAB(16)"1 Corregir este registro?"
2140 PRINT TAB(16)"2 Borrar este registro?"
2150 PRINT TAB(16)"3 Continuar sin cambios"
2160 MENU%=3:GOSUB 610:PRINT
2170 ON OP% GOSUB 2220,2280,2420
2180 PRINT CLS$;:PRINT CONTINUA$:PRINT SINO$
2190 GOSUB 690
2200 IF UPPER$(K$)="S" THEN PRINT CLS$:GOTO  2030
2210 RETURN
2220 ' ................ OPCION 1 modificar registro
2230 PRINT:F0%=1
2240 GOSUB 960
2250 F0%=0:PUT F1%
2260 PRINT
2270 RETURN
2280 ' ................... OPCION 2 borrar registro
2290 RN%=FETCHREC(F1%)
2300 GET F1%
2310 KEY$=UPPER$(LEFT$(AUTOR$,6))
2320 PRINT KEY$
2330 ERR%=DELKEY(F1%,0,0,KEY$,RN%)
2340 IF ERR%>103 THEN GOSUB 1130
2350 KEY$=UPPER$(LEFT$(TITULO$,6))
2360 ERR%=DELKEY(F1%,0,1,KEY$,RN%)
2370 IF ERR%>103 THEN GOSUB 1130
2380 KEY$=UPPER$(LEFT$(MAT$,6))
2390 ERR%=DELKEY(F1%,0,2,KEY$,RN%)
2400 IF ERR%>103 THEN GOSUB 1130
2410 RETURN
2420 RETURN
2430 PRINT
2440 PRINT"Presionar S-Siguiente, A-Anterior, C=Correcto"
2450 GOSUB 690
2460 IF UPPER$(K$)="A" THEN GOSUB 2490:GOTO 2430
```

```
2470 IF UPPER$(K$)="S" THEN GOSUB 2540 GOTO 2430
2480 RETURN
2490 ' ......... ..... . .. ...  . Opcion "Anterior"
2500 ERR%=SEEKPREV(F1%,0)
2510 IF ERR%<103 THEN GET F1%:PRINT CLS$,:GOSUB 4000
2520 PRINT IVIDEO$+"   NO HAY MAS  REGISTROS   "+NVIDEO$
2530 RETURN
2540 ' ............ ....... ..... Opcion "Siguiente"
2550 ERR%=SEEKNEXT(F1%,0)
2560 IF ERR%<103 THEN GET F1%:PRINT CLS$;.GOSUB 4000
2570  PRINT IVIDEO$+"   NO HAY MAS REGISTROS   "+NVIDEO$
2580 RETURN
3000 ' .. APERTURA DE FICHEROS Y PLANTILLA REGISTROS
3010 '
3020 OPEN "K",F1%,F$,FI$,2,LONG%,LI$
3030 FIELD F1%,20 AS AUTOR$, 50 AS TITULO$,20 AS MAT$
3040 RETURN
3050 ' ........ ...  . . .. ...   .   CERRAR FICHERO
3060 '
3070 PRINT CLS$;"CERRANDO FICHEROS"
3080 CLOSE F1%
3090 PRINT"F I N  DE LA TAREA"
3100 RETURN
3110 ' ....   .... CONTROL DE EXISTENCIA DE FICHEROS
3120 '
3130 IF ERR%=58 THEN 3150
3140 ON ERROR GOTO 0
3150 PRINT" EL FICHERO YA EXISTE '"
3160 FOR H=1 TO 3000:NEXT H
3170 RESUME NEXT
3180 END
3190 ' .. ...  .      . CONTROL DE TITULO INCORRECTO
3200 PRINT IVIDEO$
3210 PRINT"   NO EXISTE ESE TITULO '   "+NVIDEO$
3220 FOR H%=1 TO 3500:NEXT H%. ' Pausa
3230 RETURN
4000 '          . . .   .  IMPRIMIR DATOS POR AUTORES
4010 '
4020 PRINT"AUTOR   - ",AUTOR$
4030 IF PR%THEN LPRINT "AUTOR   - ",AUTOR$
4040 PRINT"TITULO   - ",TITULO$
4050 IF PR%THEN LPRINT "TITULO   - ";TITULO$
4060 PRINT"MATERIA - ",MAT$
4070 IF PR%THEN LPRINT "MATERIA - ",MAT$;CHR$(13)
4080 PRINT S$:PRINT
4090 RETURN
4100 '     . .            IMPRIMIR DATOS POR TITULOS
4110 '
4120 PRINT"TITULO  - ",TITULO$
4130 IF PR%THEN LPRINT "TITULO  - ",TITULO$
4140 PRINT"AUTOR   - ",AUTOR$
4150 IF PR%THEN LPRINT "AUTOR   - ";AUTOR$
4160 PRINT"MATERIA - ",MAT$
4170 IF PR%THEN LPRINT "MATERIA - ",MAT$,CHR$(13)
```

```
4180 PRINT S$ PRINT
4190 RETURN
4200 ´ ......  ..  .      IMPRIMIR DATOS POR MATERIAS
4210 ´
4220 PRINT"MATERIA - ",MAT$
4230 IF PR%THEN LPRINT "MATERIA - ";MAT$
4240 PRINT"AUTOR   - ";AUTOR$
4250 IF PR%THEN LPRINT "AUTOR  , - ";AUTOR$
4260 PRINT"TITULO  - ";TITULO$
4270 IF PR%THEN LPRINT "TITULO  - ";TITULO$;CHR$(13)
4280 PRINT S$·PRINT
4290 RETURN
9000 ´  .  .....  .. .. RUTINA DE BORRADO DE FICHEROS
9010 PRINT CL$+IVIDEO$+" Desea borrar todos los ficheros de
datos"
9020 PRINT" e indices utilizados por este programa? "
9030 PRINT SINO$+NVIDEO$;
9040 INPUT Z$
9050 IF UPPER$(Z$)="S" THEN 9070 ELSE 9090
9060 ERA *.DAT
9070 ERA *.DAT
9080 PRINT IVIDEO$+" FICHEROS BORRADOS "+NVIDEO$
9090 STOP
```

CAPITULO 8

FICHEROS CREADOS POR CP/M

1. INTRODUCCION

2. EL EDITOR DE CP/M

* Generalidades
* Manejo del editor. Operaciones básicas
* Otras operaciones
* Resumen de órdenes del editor ED.COM

3. UTILIZACION DE LOS FICHEROS ED DESDE BASIC

8. Ficheros creados por CP/M

1. INTRODUCCION

El sistema operativo CP/M con que cuenta su Amstrad, dispone, entre otros, de un programa editor de textos. No es propiamente un tratamiento de textos, pero sí permite introducir de forma análoga a éste, una serie de datos (textos) que serán posteriormente almacenados en un fichero. Su operatividad y funciones son muy distintas, pero bajo nuestro punto de vista —de utilización de ficheros— puede sernos igualmente útil.

EL EDITOR DE CP/M

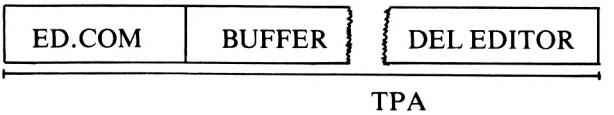
Generalidades

Antes de estudiar con detenimiento los ficheros creados por el editor de CP/M, pasemos a una somera descripción del mismo, en qué consiste el editor CP/M y cómo se utiliza —dado que no se incluye en el manual.

El programa editor es el transitorio ED.COM. Para utilizarlo debe primero entrar en CP/M y cuando obtenga la impronta A > tecle el nombre seguido de ENTER

A > ED.COM ó A > ED

El programa se instala en la base del TPA de la memoria RAM, de forma que el resto constituye el buffer del editor.



Un fichero de textos se compone de caracteres y éstos se agrupan en líneas. La separación de líneas se produce mediante dos caracteres de control: ^M y ^J que corresponden al retorno del carro y a la alimentación de línea, respectivamente. Ambos carac-

teres son generados de forma automática por ED cuando se pulsa la tecla ENTER (RETURN).

Al cargar ED observará que desaparece la impronta de CP/M, -A>- y aparece el símbolo *. Este es la impronta del editor. Su aparición indica que nos encontramos a nivel de editor, es decir, listos para trabajar con él.

El programa ED.COM se encuentra en la cara 1 del disco de sistema del CPC 6128 y la 3 de los PCW.

Para comenzar a trabajar con ED debe previamente realizar una copia de este programa en un disco debidamente formateado. Utilice la orden A>PIP y después *B: =A: ED.COM.

Dado que en el manual no se indica cómo funciona el editor, pasamos a exponer brevemente sus principales comandos. (Para mayor información puede consultar diversas obras sobre CP/M).

Nota: El símbolo ^ indica que debe presionarse la tecla CONTROL o ALT según el modelo, simultáneamente a la letra que sigue.

Manejo del editor. Operaciones básicas

Son muy variadas las funciones del programa ED. Veamos a continuación las principales.

* Llamar al editor y crear un fichero

Ya hemos visto en el apartado anterior que el programa se invoca mediante:

```
A > ED.COM
```

Pero lo más útil es simultanear la llamada con la creación (apertura) del fichero que contendrá los datos. Esto se consigue añadiendo el nombre del fichero a la orden ED, en la misma línea; es decir, si creamos un fichero de nombre PRUEBA.UNO, utilizaremos:

```
A > ED PRUEBA.UNO
```

De esta forma, se realiza simultáneamente la carga de ED.COM y la creación del archivo PRUEBA.UNO.

En pantalla obtendrá como respuesta:

```
NEW FILE
: *
```

La primera línea indica que el fichero no existía, es nuevo y la segunda es la impronta propia del editor como se ha indicado.

*** Introducir texto. Inserción**

Para comenzar a introducir textos, debe indicarse al editor que se va a insertar texto. Para ello, tras la impronta anterior, añadimos "i" seguido de ENTER, hágalo y obtendrá como respuesta:

```
: * i
1 :
```

indicando que está dispuesto para recoger la primerá línea. Introduzcamos el siguiente ejemplo. Teclee:

```
1: En          (Presione RETURN al final de cada línea)
2: un
3: lugar
4: de
5: la
6: Mancha
7:
```

Tras la introducción de la última línea, (la 6), aparecerá el indicativo 7:; es decir, se encuentra en espera de más líneas, debemos, pues, indicarle que la entrada de datos ya ha finalizado, saliendo del modo inserción en que nos encontramos.

*** Salir del modo inserción**

Para abandonar el modo inserción, basta con presionar Z (= CONTROL o ALT, según modelo). Tras ello volverá a reaparecer la impronta del editor:

haga 7:^Z

y la pantalla quedará en la forma siguiente:

```
:
:
6: Mancha
7:
:*
```

*** Grabar el fichero**

Puesto que hemos terminado con la introducción de líneas, procedemos a grabar su contenido en el fichero que fue creado previamente al efecto.

La orden de grabación es:

```
: * e
```

Al pulsar ENTER verá que el disco es activado y en la pantalla volverá a reaparecer la impronta de CP/M, A >. El control ya no está bajo ED, sino bajo el sistema operativo.

Utilice la orden DIR y obtendrá:

```
ED      .COM
PRUEBA.UNO
PRUEBA.BAK
```

El programa PRUEBA.BAK tiene su origen en un fichero transitorio creado por la orden ED PRUEBA.UNO, que crea automáticamente el fichero PRUEBA.$$$ que posteriormente pasará a consolidarse como PRUEBA.UNO, por la orden de grabación "e", por lo que PRUEBA.$$$ pasa a ser PRUEBA.BAK (duplicado).

Si lo desea puede listar el contenido del fichero PRUEBA, mediante la orden permanente TYPE (CP/M).

```
A > TYPE PRUEBA.UNO
```

Otras operaciones

Las operaciones y funciones del editor ED.COM de CP/M son muy variadas y su estudio escapa de la finalidad práctica de este libro. Por ello, remitimos al lector a la bibliografía específica para un estudio profundo sobre este editor.
No obstante, vamos a ver a continuación algunas órdenes complementarias, que permiten la inserción y borrado —fundamentales para trabajar con ED y que le permitirá crear ficheros como los indicados en los capítulos 0 y 1.

* Inserción de líneas

Partiendo del fichero creado anteriormente y guardado en disco bajo el nombre PRUEBA.UNO, procedemos nuevamente a llamar al editor indicándole el nombre del fichero en cuestión:

```
A > ED PRUEBA.UNO
: *
```

Dado que el fichero ahora sí existe, no aparece el mensaje NEW FILE. Pero esta orden, por sí sola, no carga el contenido del fichero en el buffer del editor, para ello hemos de añadir:

```
: * # a
```

Al presionar ENTER se realiza la transferencia disco-buffer ED, tras la cual aparece:

```
1: *
```

indicando que la orden ha sido ejecutada, aun cuando no se muestre el texto en pantalla. Para visualizarlo se utiliza la orden # t (tipografiar), tras la cual, en pantalla se obtiene:

```
A > ED PRUEBA.UNO
: * # a
1: * #t
1: En
2: un
3: lugar
4: de
5: la
6: Mancha
1: *
```

La expresión 1: * que figura en último lugar, es indicativa de que cualquier inserción sería colocada en primer lugar, es decir, antes de la línea "En". Probemos:

```
1: * i
1: D. QUIJOTE
2:
```

Para salir del modo de inserción puede utilizarse ^Z, que no produce grabación. Si se desea grabar el nuevo fichero basta con usar tras ^Z, la orden e como se hizo anteriormente. Si lo hace puede comprobar con TYPE que el fichero tiene una línea más, precisamente la que se insertó en primer lugar, de forma que el número de línea se ha desplazado.

Si lista el fichero obtendrá

```
A >ED PRUEBA.UNO
: * # a                    (pasa texto al buffer)
1: # t                     (lista texto)
1: D. QUIJOTE
2: En
3: un
4: lugar
5: de
6: la
7: Mancha
```

La tarea de inserción puede realizarse en cualquier parte del texto, p.e. si entre la línea 3 y 4 se desea insertar "extenso" el procedimiento será

1) orden i para insertar
2) 4: para indicar que la inserción se realizará delante de la línea 4
3) 4: extenso. Introducir el texto tras la impronta 4: dada por el editor
4) Z para salir de inserción

Los errores mecanográficos al escribir las líneas, pueden ser subsanados mediante ^H, que es la orden específica de borrado de CP/M, cuyo efecto es eliminar el carácter situado a la izquierda del cursor. En el epígrafe siguiente se describen con más detalle las posibilidades de borrado y el movimiento de los cursores.

*** Borrado parcial de líneas. Movimiento de los cursores**

El movimiento sobre el texto, o sobre una línea no es precisamente una tarea cómoda en ED, por lo que en muchas ocasiones resulta preferible borrar una línea y escribirla de nuevo, que tratar de modificarla.

Para poder actuar sobre una determinada línea, debemos previamente, situar el puntero en dicha línea, valiéndonos para ello de la orden n:, siendo n el número de línea en cuestión.

Una vez situados, ED dispone de una orden específica de borrado parcial de líneas o conjunto de caracteres. Su sintaxis es la siguiente:

: ± n d

Donde n indica el número de caracteres a suprimir, y la letra d es la inicial de delete (borrar). El signo + o - indica la dirección en que se va a realizar el borrado; (-) a la izquierda del cursor, (+) a la derecha. Si se omite el signo, se toma (+) por defecto.

Ahora bien, además de la orden n: para el desplazamiento de líneas, ED dispone de un juego de órdenes que permiten desplazamientos de los cursors en los cuatro sentidos: arriba-abajo (líneas), derecha-izquierda (en una línea). Veamos estas órdenes:

: ± b

b es la inicial de begin o bottom (comienzo o fondo). La función asumida es situar el puntero al principio o al final del fichero, según que el signo sea - o +, respectivamente. Por omisión se toma el signo +:

: ± n l

l es la inicial de line (línea). Esta orden permite avanzar o retroceder un número de líneas determinado por el número n, según el signo empleado, como en la orden anterior.

El número n puede ser sustituido por el comodín #, que equivale a "todo", de manera que: # l hace avanzar al puntero hasta el final y: — # l, al principio del texto.

: ± n c

c es la inicial de character y la función asumida es mover el cursor a lo largo de la línea el número de caracteres indicado por n. El signo - indica hacia la izquierda y el + hacia la derecha (por omisión se toma +).

Para ver un funcionamiento práctico basándonos en el ejemplo que hemos venido utilizando, imaginemos que deseamos modificar la línea 3 que contiene:

3: lugar

y en vez de "lugar" poner "local".

Las operaciones a realizar serán por este orden:

— Llevar el puntero al comienzo : —b
— Llevar puntero a línea 3 : 3 l
— Llevar puntero a la letra l : l c
— Borrar "ugar" : 4 d
— Añadir "ocal" : i ocal

Como se ve no resulta precisamente cómodo, veamos como borrar líneas.

*** Borrado de líneas completas**

Para borrar una o varias líneas consecutivas, ED dispone de la orden:

: ± n k

La letra k es la inicial de kill (suprimir) y n es el número de líneas a borrar, por delante o por detrás de la línea en curso (puntero) según el signo (-arriba, + abajo).

Por ejemplo para borrar la línea 3 completa, habrá que dar las siguientes órdenes:

1: * 3: posicionar puntero en línea 3
3: * k borrar línea 3
3: * b ir al final del fichero
l: * # t listar todo

Las órdenes de ED pueden concatenarse, de forma que el ejemplo de borrado anterior, también puede realizarse mediante:

1: * 3 k b # t

Cuando el valor de n es uno, puede omitirse.

El uso del comodín permite borrar todas las líneas anteriores o posteriores a la posición del puntero. La sintaxis será:

: ± # k

*** Conclusión de los trabajos**

Además de la orden ^C general de CP/M que produce una interrupción brusca de cualquier trabajo, ED contiene una orden propia de salida, mucho más aconsejable y útil que la anterior. Esta orden es "q", inicial que quit (desistir).

Tras la introducción de esta orden, ED pide confirmación:

```
: *q
Q - (Y/N)?
```

Si se pulsa Y, las modificaciones efectuadas son abandonadas. Si se pulsa N, la situación se mantiene inalterada, pasando a nivel de operador.

Una orden complementaria es "o" inicial de original, que deja el fichero en las condiciones originales, es decir, no se tienen en cuenta las modificaciones.

Al igual que la orden "q", tras la utilización de "o", ED pide confirmación:

```
: * o
O - (Y/N)?
```

El efecto es el mismo que con "q", pero se mantiene el nivel de editor, en vez de pasar a nivel de operador.

Las órdenes del ED expuestas son sólo algunas de las utilizables con este programa, aunque estimamos suficientes para la creación y corrección de ficheros.

Resumen de órdenes del editor ED.COM

ED nombre	Carga el programa ED.COM y crea el fichero "nombre" si no existía, en caso contrario lo abre.
i	Insertar texto. Una o varias líneas.
# a	Transfiere el texto desde un fichero al buffer.
# t	Lista el fichero contenido en el buffer por pantalla e impresora, si ésta se encuentra activada por P̂.
e	Transfiere el contenido del buffer al disco, almacenándolo en el fichero que se indicó.
n:	Inserta una línea en la posición indicada por el número n.
# a # t	Realiza simultáneamente las funciones de # a y # t.
± n d	Borrar el número de caracteres indicado por n.
± b	Puntero al principio o final del fichero.
± n l	Avanzar o retroceder n líneas.
± n c	Avanzar o retroceder n caracteres en una línea.

±n k	Borrar n líneas.
#	Comodín cuyo significado es "todo".
q	Salida a nivel de operador.
o	Salida manteniéndose a nivel de editor.

3. UTILIZACION DE LOS FICHEROS ED DESDE BASIC

Los ficheros creados por ED.COM, pueden ser leídos y manipulados desde Mallard BASIC, como ficheros secuenciales.

Para poder comprobarlo puede utilizarse la orden TYPE. Haga bajo CP/M:

A > TYPE PRUEBA.UNO

y obtendrá un listado del contenido del fichero línea a línea.

Ahora procedamos a su lectura y comparación mediante el siguiente programa BASIC.

```
10 ' LECTURA DE FICHEROS CREADOS POR ED.COM
20 ' ............... "ED" ..................
30 INPUT "NOMBRE DEL FICHERO ";F$
40 OPEN "I",1,F$
50 WHILE NOT EOF (1)
60 INPUT .1,A$:PRINT A$
70 WEND
80 CLOSE
```

De igual forma puede establecerse una correspondencia entre ficheros LocoScript-BASIC-CP/M, vea el programa incluido en el apéndice 5, que trata del paso de un fichero creado por Loco Script a uno estandar ASCII.

Los ficheros PROFILE.SUB indicados en los capítulos 0 y 1, pueden ser confecciones por ED directamente, o bien con LocoScript y convertidos posteriormente a ASCII, siguiendo las instrucciones indicadas en el citado apéndice.

APENDICES

1. CODIGO ASCII
2. LA IMPRESORA DE LOS PCW
3. CODIGOS DE ERRORES EN MALLARD BASIC
4. CONVERSION DE NUMEROS. CAMBIO DE BASE
5. CONVERSION DE FICHEROS LOCOSCRIPT A ASCII
6. CODIGOS DE CONTROL
7. EDITOR SENCILLO PARA FICHEROS DE ORDENES

Apéndice 1

CODIGO ASCII

SIM	DEC	HEX	BINARIO	COMENTARIO
NUL	0	00	00000000	NULO [C-@]
SQH	1	01	00000001	COMIENZO DE CABECERA [C-A]
STX	2	02	00000010	COMIENZO DE TEXTO [C-B]
ETX	3	03	00000011	FIN DE TEXTO [C-C]
EOT	4	04	00000100	FIN DE TRANSMISION [C-D]
ENQ	5	05	00000101	PETICION [C-E]
ACK	6	06	00000110	RECONOCIMIENTO [C-F]
EEL	7	07	00000111	ALARMA O CAMPANA [C-G]
BS	8	08	00001000	ESPACIO HACIA ATRAS [C-H]
HT	9	09	00001001	TABULACION HORIZONTAL [C-I]
LF	10	0A	00001010	ALIMENTACION DE LINEA [C-J]
VT	11	0B	00001011	TABULACION VERTICAL [C-K]
FF	12	0C	00001100	ALIMENTACION DE PAGINA [C-L]
CR	13	0D	00001101	RETORNO DEL CARRO [C-M]
SO	14	0E	00001110	DESPLAZAMIENTO EXTERNO [C-N]
SI	15	0F	00001111	DESPLAZAMIENTO INTERNO [C-O]
DLE	16	10	00010000	ESCAPE DEL ENLACE DE COMUNICACION DE DATOS [C-P]
DC1	17	11	00010001	CONTROL UNIDAD 1 [C-Q]
DC2	18	12	00010010	CONTROL UNIDAD 2 [C-R]
DC3	19	13	00010011	CONTROL UNIDAD 3 [C-S]
DC4	20	14	00010100	CONTROL UNIDAD 4 [C-T]
NAK	21	15	00010101	RECONOCIMIENTO NEGATIVO [C-U]
SYN	22	16	00010110	DOSECUPACION SINCRONA [C-V]
ETB	23	17	00010111	FIN DE TRANSMISION DE BLOQUE [C-W]
CAN	24	18	00011000	CANCELACION [C-X]
EM	25	19	00011001	FIN DE MEDIO [C-Y]
SUB	26	1A	00011010	SUSTITUCION [C-Z]
ESC	27	1B	00011011	ESCAPE [C-[]
FS	28	1C	00011100	SEPARADOR DE FICHERO [C-\]
GS	29	1D	00011101	SEPARADOR DE FRUPO [C-]]
RS	30	1E	00011110	SEPARADOR DE REGISTRO [C-^]
SS	31	1F	00011111	SEPARADOR DE UNIDAD [C-O]
SP	32	20	00100000	ESPACIO
!	33	21	00100001	SIMBOLOS
	34	22	00100010	
#	35	23	00100011	
$	36	24	00100100	
%	37	25	00100101	
&	38	26	00100110	
'	39	27	00100111	
(	40	28	00101000	
)	41	29	00101001	
*	42	2A	00101010	

SIM	DEC	HEX	BINARIO	COMENTARIO
+	43	2B	00101011	
,	44	2C	00101100	
-	45	2D	00101101	
	46	2E	00101110	
/	47	2F	00101111	
0	48	30	00110000	NUMEROS
1	49	31	00110001	
2	50	32	00110010	
3	51	33	00110011	
4	52	34	00110100	
5	53	35	00110101	
6	54	36	00110110	
7	55	37	00110111	
8	56	38	00111000	
9	57	39	00111001	
:	58	3A	00111010	
;	59	3B	00111011	
<	60	3C	00111100	
=	61	3D	00111101	
>	62	3E	00111110	
?	63	3F	00111111	
@	64	40	01000000	
A	65	41	01000001	LETRAS MAYUSCULAS
B	66	42	01000010	
C	67	43	01000011	
D	68	44	01000100	
E	69	45	01000101	
F	70	46	01000110	
G	71	47	01000111	
H	72	48	01001000	
I	73	49	01001001	
J	74	4A	01001010	
K	75	4B	01001011	
L	76	4C	01001100	
M	77	4D	01001101	
N	78	4E	01001110	
O	79	4F	01001111	
P	80	50	01010000	
Q	81	51	01010001	
R	82	52	01010010	
S	83	53	01010011	
T	84	54	01010100	
U	85	55	01010101	
V	86	56	01010110	
W	87	57	01010111	
X	88	58	01011000	
Y	89	59	01011001	
Z	90	5A	01011010	
[	91	5B	01011011	SIMBOLOS
\	92	5C	01011100	
]	93	5D	01011101	
^	94	5E	01011110	
_	95	5F	01011111	
`	96	60	01100000	
a	97	61	01100001	LETRAS MINUSCULAS
b	98	62	01100010	
c	99	63	01100011	

SIM	DEC	HEX	BINARIO	COMENTARIO
d	100	64	01100100	
e	101	65	01100101	
f	102	66	01100110	
g	103	67	01100111	
h	104	68	01101000	
i	105	69	01101001	
j	106	6A	01101010	
k	107	6B	01101011	
l	108	6C	01101100	
m	109	6D	01101101	
n	110	6E	01101110	
o	111	6F	01101111	
p	112	70	01110000	
q	113	71	01110001	
r	114	72	01110010	
s	115	73	01110011	
t	116	74	01110100	
u	117	75	01110101	
v	118	76	01110110	
w	119	77	01110111	
x	120	78	01111000	
y	121	79	01111001	
z	122	7A	01111010	
{	123	7B	01111011	SIMBOLOS
|	124	7C	01111100	
}	125	7D	01111101	
	126	7E	01111110	

Apéndice 2. La Impresora de los PCW

El control de la impresora integrada de los PCW, desde Mallard BASIC, se realiza mediante la instrucción LPRINT, seguida de códigos y caracteres de control.

Los códigos de control se introducen mediante la orden CHR$(cc), donde cc representa el valor ASCII de dicho código, en decimal o hexadecimal. Por ejemplo el tipo de letra "estrecha" se consigue enviando a la impresora la sentencia

```
LPRINT CHR$(15)
```

en decimal, o bien

```
LPRINT CHR$(&H0F)
```

en hexadecimal

En la tabla de código ASCII que se incluye en el apéndice A1, puede verse la equivalencia entre valores decimales y hexadecimales, así como los símbolos a que corresponden, dichos símbolos no pueden ser usados directamente desde BASIC, pero si en CP/M para la creación de ficheros de definición de la impresora, en el ejemplo anterior, el símbolo que corresponde al valor 15 decimal es SI.

En cuanto a los caracteres de control, se envían a la impresora mediante una orden LPRINT, en la cual van entrecomillados. Por ejemplo fijar el margen izquierdo a 5 caracteres del límite de escritura, supone enviar los caracteres ESC l 5.

ESC— símbolo ASCII cuyo valor es 27 (decimal)
l— carácter de control a enviar en la forma "l" (ele minúscula)
5— posición del margen izquierdo

la sentencia sería

```
LPRINT CHR$(27);"l"; CHR$(5) o bien
LPRINT CHR$(27); CHR$(108); CHR$(5)
```

dado que el valor ASCII de l (ele minúscula) es 108 decimal.

Habida cuenta de la variedad de formas en que pueden introducirse los códigos y a fin de minimizar errores, lo más aconsejable es decidirse por un tipo de numeración —decimal—, un tipo de representación de caracteres —también decimal— y un tipo

de símbolos —como no! decimal—. Esta forma tiene la ventaja de uniformizar todos los tipos. En todos los controles de impresora que se describen a continuación se utilizará la grafía ordinaria y su equivalencia en decimal.

Para mayor comodidad en la introducción de las palabras BASIC, LPRINT y CHR$, puede utilizarse la rutina de definición de teclado, descrita en el apéndice A.3.

En el cuadro CU.1 se exponen agrupados los códigos de control de la impresora. La primera columna contiene las funciones que realizan. En la segunda la nomenclatura de los códigos, expresada en símbolos ASCII, válida para el trabajo en CP/M, pero no en BASIC. La tercera contiene los códigos anteriores en forma utilizable mediante LPRINT y por último la cuarta expresado todo en valores decimales y por tanto utilizable también por LPRINT del BASIC, unido a la función CHR$ que contendrá los valores expuestos en dicha columna, tal como se hace en el ejemplo anterior.

CU.1 CODIGOS DE LA IMPRESORA

FUNCION	SIMBOLOS DE CONTROL ASCII	LPRINT	Valores decimales CHR$( )
TIPO DE PAPEL			
Hojas sueltas	ESC $	CHR$(27);"$"	(27) (36)
Papel continuo	ESC c	CHR$(27);"c"	(27) (99)
AVANCE DEL PAPEL			
Salto de página	FF	CHR$(12)	(12)
Sensor fin de papel activado	ESC 8	CHR$(27);"8"	(27) (56)
id. desactivado	ESC 9	CHR$(27);"9"	(27) (57)
AVANCE DE LINEAS			
Avance de línea	LF	CHR$(10)	(10)
Activar avance automático	ESC LF	CHR$(27);"LF"	(27) (10)
Desactivar avance automático	ESC CR	CHR$(27);"CR"	(27) (13)
SALTO FIN DE PAGINA			
Activar (n líneas)	ESC N n	CHR$(27);"N";CHR$(n)	(27) (78) (n)
Desactivar	ESC O	CHR$(27);"O"	(27) (79)
CARRO			
Retroceso de 1 posición	BS	CHR$(8)	(8)
Retorno a principio de línea	CR	CHR$(13)	(13)
PASOS DE LINEA			
1/8 de pulgada	ESC 0	CHR$(27);"0"	(27) (48)
7/72 " "	ESC 1	CHR$(27);"1"	(27) (49)

FUNCION	SIMBOLOS DE CONTROL ASCII	LPRINT	Valores decimales CHR$( )
1/6 " "	ESC 2	CHR$(27);"2"	(27) (50)
n/256 " "	ESC 3 n	CHR$(27);"3";CHR$(n)	(27) (51) (n)
n/72 " "	ESC A n	CHR$(27);"A";CHR$(n)	(27) (65) (n)
MARGENES (ver CU.2)			
Izquierdo	ESC l n	CHR$(27);"l";CHR$(n)	(27) (108) (n)
Derecho	ESC Q n	CHR$(27);"Q";CHR$(n)	(27) (81) (n)
TABULADORES (Max 32)			
Marcar posición tabulador	ESC D n	CHR$(27);"D";CHR$(n)	(27) (68) (n)
Avanzar hasta tabulador	HT	CHR$(9)	(9)
LONGITUD DE PAGINA			
Definición en líneas	ESC C n	CHR$(27);"C";CHR$(n)	(27) (67) (n)
Definición en pulgadas	ESC C 0 n	CHR$(27);"C";∅;CHR$(n)	(27) (67) (48) (n)
REINICIALIZAR			
Volver a condición inicial	ESC @	CHR$(27);"@"	(27) (64)
JUEGO DE IDIOMAS			
EE.UU.	ESC R 0	CHR$(27);"R";CHR$(0)	(27) (82) (0)
Francia	1	' ' (1)	(1)
Alemania	' 2	' ' (2)	(2)
Reino Unido	' 3	' ' (3)	(3)
Dinamarca	' 4	' ' (4)	(4)
Suecia	' 5	' ' (5)	(5)
Italia	' 6	' ' (6)	(6)
España	' 7	' ' (7)	(7)
Japón	' 8	' ' (8)	(8)

Los valores que pueden tomar "l" y "Q" en la delimitación de los margen izquierdo y derecho respectivamente, vienen impuestos por el tipo de letra, que fija el número de caracteres máximo que puede contener una línea.

En el cuadro CU.2 se indica, para cada tipo de letra, los valores posibles de "l" y "Q", así como los valores implícitos al encender la impresora, por último se expresa la distancia mínima entre el tabulador izquierdo y derecho, expresada en caracteres.

CU.2 MARGENES. VALORES DE DELIMITACION E IMPLICITOS

TIPO DE LETRA	DELIMITACION		VAL.IMPLI.		DIST. MINIMA
	IZQ	DRCHA	IZQ	DRCHA	ENTRE TAB
PICA (10 cpp)	0-78	2-81	0	80	2 (c)
PICA DOBLE ANCHO (5 cpp)	0-39	1-40	0	40	1 (c)
ELITE (12 cpp)	0-93	3-96	0	96	2 (c)
ELITE DOBLE ANCHO (6 cpp)	0-46	2-48	0	48	1 (c)
PROPORCIONAL (12 cpp)	0-93	3-96	0	96	2 (c)
ESTRECHA (17 cpp)	0-133	4-139	0	137	3 (c)
ESTREC. DOBLE ANCHO (8.5cpp)	0-66	2-69	0	68	1 (c)
DISTANCIA MINIMA ENTRE TABULADORES = 1/5 pulgada					

(c) = caracteres (cpp) = Número de caracteres por pulgada
1 pulgada = 2.54 cm

Los códigos de los distintos tipos de letra se exponen a continuación. Por razones de espacio se ha suprimido la columna intermedia equivalente al cuadro CU.1, sin embargo se indican los códigos que activan y desactivan cada tipo de letra, tenga en cuenta que la desactivación supone: la vuelta al tipo de letra "Pica" que es la adoptada al encender o reinicializar la impresora o bien el abandono de un atributo (negrilla, alta calidad), etc.

TIPOS DE LETRA

LETRA	ACTIVAR		DESACTIVAR	
	ASCII	Valores decimales	ASCII	Valores decimales
Estrecha	SI	(15)	DC2	(18)
Elite	ESC M	(27) (77)	ESC p	(27) (112)
Proporcional	ESC p 1	(27) (112) (1)	ESC p 0	(27) (112) (0)
Doble ancho	SO	(14)	DC 4	(20)
Cursiva	ESC 4	(127) (52)	ESC 5	(27) (53)
Alta calidad	ESC m 1	(27) (109) (1)	ESC m 0	(27) (109) (0)
Doble impresión	ESC G	(27) (71)	ESC H	(27) (72)
Negrita	ESC E	(27) (69)	ESC F	(27) (70)
Subíndices	ESC S 1	(27) (83) (1)	ESC T	(27) (84)
Superíndices	ESC S 0	(27) (83) (0)	ESC T	(27) (84)
Subrayado	ESC - 1	(27) (45) (1)	ESC - 0	(27) (45) (0)

Existe un carácter especial ESC!n que permite seleccionar un modo mixto de escritura, p.e. dobla ancha y negrilla. El número n representa el valor decimal de un número binario de 5 dígitos (bits), con el siguiente significado.

bit 1° — doble ancho (bit más significativo)
bit 2° — doble impresión
bit 3° — negrilla
bit 4° — estrecha
bit 5° — élite (bit menos significativo)

Si tenemos en cuenta que el valor decimal de cada bit es

bit.	1°	2°	3°	4°	5°
valor dec.	16	8	4	2	1

Puede establecerse la siguiente tabla de valores decimales

16 — doble ancho
8 — doble impresión
4 — negrilla
2 — estrecha
1 — élite

Para elegir una combinación y calcular el valor de n correspondiente basta con sumar los valores de cada opción, por ejemplo:

doble ancho y negrilla	n= 16 + 4 = 20	(10100)
élite y doble impresión	n= 1 + 8 = 9	(01001)

Este método es más sencillo que establecer el binario correspondiente y pasarlo a decimal.

El cero puede ser representado por la impresora con o sin barra, se selecciona mediante

ESC X	cero barrado	CHR$(27);"X"	(27) (88)
ESC o	cero sin barra	CHR$(27);"o"	(27) (111)

El juego de caracteres normal puede ser ampliado entre los márgenes 0 a 31 y 128 a 159, siempre que dichos caracteres no sean de control. Para activar esta ampliación utilice ESC I 1 y para desactivar ESC I 0, en el manual (tomo 1) se indica una tabla con los caracteres ampliados.

Dado que el envío de caracteres de control a la impresora resulta siempre tedioso, por las continuas cadenas LPRINT CHR$( )+..., pueden utilizarse varios tipos de simplificaciones:

1ª. Redefinir el teclado de forma que "CHR$(" quede incluido en una tecla y su sola pulsación genera este conjunto de caracteres. Esta posibilidad fue ofrecida en los capítulos 0 y 1, instalación.

2ª. Definir cadenas con la orden DEF FN, por ejemplo

10 DEF FN I$(X) = CHR$(27) + CHR$(X)

de esta forma podemos utilizar FN para imprimir

15 LPRINT FN I$(14) + "EJEMPLO CON LETRA EXPANDIDA"

3ª. Definir una variable alfanumérica con la secuencia de escape CHR$(27)

E$ = "CHR$(27)"

4ª. Elaborar una rutina de impresora con las secuencias de códigos de control más utilizadas y agregarla a todos los programas. Cada conjunto de caracteres de control se le asigna a una variable alfanumérica cuyo nombre sea fácil de recordar e identificar.

El listado que se ofrece a continuación, que puede ser ampliado, mejorado y/o adaptado a sus necesidades permite este tipo de simplificación.

```
1000 ' RUTINA DEFINICION CODIGOS IMPRESORA
1010 '............... IMPRE ...............
1020 '     A=Activar      D=Desactivar
1030 ' LETRA ELITE
1040 AELITE$=CHR$(27)+CHR$(77)
1050 DELITE$=CHR$(27)+CHR$(112)
1060 ' LETRA ESTRECHA
1070 AESTRE$=CHR$(27)+CHR$(15)
1080 DESTRE$=CHR$(27)+CHR$(18)
1090 ' LETRA PROPORCIONAL
1100 APROPOR$=CHR$(27)+CHR$(112)+CHR$(1)
1110 DPROPOR$=CHR$(27)+CHR$(112)+CHR$(0)
1120 ' LETRA EXPANDIDA (Doble ancho)
1130 AEXPAN$=CHR$(27)+CHR$(14)
1140 DEXPAN$=CHR$(27)+CHR$(20)
1150 ' LETRA ITALICA (Inclinada)
1160 AITALI$=CHR$(27)+"4"
1170 DITALI$=CHR$(27)+"5"
1180 ' LETRA NEGRITA
1190 ANEGRA$=CHR$(27)+CHR$(69)
1200 DNEGRA$=CHR$(27)+CHR$(70)
1210 ' LETRA ALTA CALIDAD
1220 AACALI$=CHR$(27)+CHR$(109)+CHR$(1)
```

```
1230 DACALI$=CHR$(27)+CHR$(109)+CHR$(0)
1240 ´ SUBRAYADO
1250 ASUBRA$=CHR$(27)+"-"+"1"
1260 DSUBRA$=CHR$(27)+"-"+"0"
1270 ´ INICIALIZAR IMPRESORA
1280 INIT$=CHR$(27)+CHR$(64)
1290 ´ SALTOS DE LINEA Y PAGINA
1300 SALTOPAG$=CHR$(12)
1310 SALTOLIN$=CHR$(13)
```

Una vez instalada la rutina expuesta, en su programa, para provocar el envío de un conjunto de caracteres de control a la impresora bastará con imprimir la variable que los contiene, por ejemplo para enviar la orden de escritura con tipo itálico sería

```
LPRINT AITALI$;
```

El punto y coma evita el que se produzca un salto de línea. También puede añadirse a la anterior secuencia la variable que contenga el texto a imprimir en la forma

```
LPRINT AITALI$;Z$
```

o bien el propio texto en forma de cadena, por ejemplo

```
LPRINT AESTRE$ + "EJEMPLO DE ESCRITURA CON LETRA ESTRECHA"
```

Apéndice 3. Código de Errores de Mallard Basic

En el Apéndice II del Manual (Tomo 2), encontrará una lista exhaustiva de los mensajes de error emitidos por Mallard BASIC. Como verá, se halla dividida en tres partes, la primera, errores 1 a 30, contiene los errores denominados "ordinarios", propios de la sintaxis y actuación del lenguaje. La parte segunda, errores 50 a 74, son propios del manejo de disco y por último, la tercera, errores 113 a 115 incluye los originados en trabajos con ficheros de acceso por claves.

Código de Error	Tipo de Error
1 - 30	Ordinarios del BASIC
50 - 74	De disco
113-111	Jetsam

No se trata pues de repetirlos uno a uno en este apéndice, sino de hacer un uso más avanzado de los mismos a través de los códigos que generan y aprovechando las funciones de control de error, de que dispone el Mallard BASIC.

ERR	Esta función devuelve el código del error detectado
ERL	Indica la línea en la que se ha producido el error
ERROR	Provoca una acción idéntica a la que se realizaría si se hubiese detectado el número de error indicado.
OSERR	Permite puntualizar el genérico error 21, propio del sistema operativo.

Todas estas funciones pueden ser tratadas mediante la orden ON ERROR, elaborando una rutina informativa sobre los errores producidos durante la ejecución de un programa, de forma que éste no se detenga pero avise si se producen errores.

Una forma práctica, entre las muchas posibles, de utilizar estos códigos es la de devolver el control del programa al Menú principal tras la aparición de un error, indicando cual es el error producido y en que lugar. Para ello debe incluirse una senten-

cia al comienzo del programa principal del tipo ON ERROR. Un ejemplo puede ser:

```
.
.
.
40 ON ERROR GOTO 9000
.
.
.
100 MENU PRINCIPAL
.
.
.
9000' CONTROL DE ERRORES
9010 IF ERR<31 THEN T$= "ORDINARIO "ELSE T$ = "DE DISCO"
9020 PRINT "ERROR"; T$; "NO"; ERR; "EN LINEA"; ERL
9030 GOTO 100
```

Este tipo de rutinas puede evidentemente ser más complejas. El programador debe construir una a su gusto y necesidades, incluyéndola en todos sus programas.

Recuerde que ON ERROR en combinación con RESUME le permite reanudar la ejecución de un programa, tras la aparición de un error, en la forma que le sea más conveniente.

RESUME. Devuelve el control al principio de la sentencia donde se generó el

RESUME nº línea. Continúa el programa en la línea indicada (sustituye a GOTO 100 del ejemplo)

RESUME NEXT. Continúa el programa en la siguiente línea a la que se detectó el error.

Apéndice 4. Conversión de Números. Cambio de Base

Con el fin de facilitar la conversión de números de una base de numeración a otra, se incluye el siguiente programa, válido para cualquier base de origen y final comprendida entre 2 y 35.

El programa solicita la base en la que está expresado el número a transformar (base de partida), la base en la que se deben expresar dicho número (base final) y por último el número en cuestión.

```
10   SISTEMAS DE NUMERACION  CAMBIO DE BASES
15                   BASES
20 Z$="0123456789ABCDEFGHIJKLMNOPQRSTUVXYZ"
30 PRINT CHR$(27)+"E"+CHR$(27)+"H"
40 INPUT "BASE DE PARTIDA (MAX=35) ",B1 B1=INT(B1)
50 PRINT
60 INPUT "BASE FINAL       (MAX 35) ",B2 B2=INT(B2)
70 PRINT "NUMERO A TRANSFORMAR DE BASE",B1,"A BASE",B2,
80 INPUT N$
90 A=0 M$=""
100 L=LEN(N$)
110 FOR X=1 TO L
120 FOR Y=1 TO B1
130 IF MID$(Z$,Y,1)<>MID$(N$,X,1) THEN 150
140 Z0=Z0+INT((Y-1)*(B1^(L-X))+0 5)
150 NEXT Y
160 NEXT X
170 Z1=INT(B2*((Z0/B2)-INT(Z0/B2))+1 5)
180 M$=MID$(Z$,Z1,1)+M$
190 Z0=INT(Z0/B2)
200 IF Z0>0 THEN 170
210 PRINT PRINT
220 PRINT N$,"  en base ",b1,"  es igual a ",m$,"  en base ",b2
230 PRINT"------------------------------------------------------------"
240 PRINT PRINT PRINT PRINT
250 INPUT "OTRO NUMERO CON LAS MISMAS BASES (S/N)",C$
260 IF UPPER$(C$)="S" THEN 70
270 GOTO 30
```

Apéndice 5. Conversión de Ficheros LocoScript a ASCII

Los ficheros generados por LocoScript no tienen formato ASCII y por tanto no son utilizables por CP/M. Sin embargo el uso de este programa es mucho más cómodo que el editor de CP/M, ED.COM, por tanto resulta de gran interés el poder pasar de un formato propio como es el LocoScript a uno estandar como ASCII. Esta operación es posible a través del programa en Mallard BASIC, incluído al final de este apéndice.

Antes de utilizar un fichero de LocoScript en el programa mencionado, es necesario eliminar la cabecera del fichero, para ello puede crearse un nuevo documento sin cabecera, mediante la opción f8. Este fichero será tratado por LocoScript como un documento ajeno a él y debe ser grabado en el grupo 0, exclusivamente.

Una vez salvado el documento, como bloque, debe salir de LocoScript y pasar a CP/M para cargar Mallard BASIC, o bien utilizar la instalación descrita en el capítulo primero, si dispone ya de ella. Ejecute el programa, que se incluye a continuación y el texto LocoScript, pasará a texto ASCII.

```
10' CONVERSION DE FICHERO
20' LOCOSCRIPT - ASCII
30' ---- "LOCO ASCI"
50 PRINT CHR$(27) + "E" + CHR$(27) + "H"
60 INPUT "Nombre fichero LOCOSCRIPT"; LOCO$
70 INPUT "Nombre fichero ASCII"; ASCI$
80 IF ASCI$ = " " THEN ASCI$ = LOCO$
90 PRINT "Fichero Fuente"; LOCO$; "Fichero destino"; ASCI$
100 INPUT "Nombres correctos (S/N)"; Z$
110 IF UPPER$(Z$) < > "S" THEN 50
120 OPEN "I", 1, LOCO$: OPEN "O", 2, ASCI$
130 FOR K = 1 TO 128: T$ = INPUT$ (1, # 1): NEXT K
140 WHILE NOT EOF (1)
150 T$ = INPUT$(1, # 1)
160 IF T$ = CHR$(129) THEN T$ = " "
170 IF T$ = CHR$(136) THEN T$ = INPUT$(1, # 1): A$ = CHR$(13) +
  CHR$(10)
180 IF T$ < > CHR$(128) THEN PRINT # 2, T$;
```

```
190 WEND
130 CLOSE
150 PRINT "Fichero modificado a ASCII"
160 END
```

Como verá la línea 130 lee, despreciando el contenido, de la sección primera del fichero que contiene datos de uso exclusivo de LocoScript, pero que no pertenecen al texto propiamente dicho.

Las líneas 160 a 180 manejan los retornos de carro, para adaptarlos al standar ASCII.

El fichero generado puede ser usado desde CP/M y desde BASIC. Una aplicación inmediata es la creación de ficheros PROFILE.SUB descrita en los capítulos 0 y 1, utilizando LocoScript, en vez de ED.COM, ya que si no está familiarizado con este editor, le será mucho más fácil el procedimiento descrito en este apéndice.

Apéndice 6. Códigos de Control

SECUENCIAS DE ESCAPE

A título de referencia se incluyen a continuación todas las secuencias de escape utilizables por los CPC y PCW con indicación de las diferencias entre ambos modelos en aquellos casos en que existen.

— ESC "0" y ESC "1". Desactiva y activa respectivamente la línea de mensajes del sistema operativo. Su desactivación permite utilizar dicha línea en la forma habitual, los mensajes no se pierden sino que aparecen junto al resto de los textos, como uno más.

— ESC "2" n. Permite modificar el juego de caracteres internacional. El valor de n será entero comprendido entre 0 y 7 con el siguiente significado.

0 U.S.A.	4 Dinamarca
1 Francia	5 Suecia
2 Alemania	6 Italia
3 Reino Unido	7 España

— ESC "3" n. Modifica el modo de pantalla en los CPC. En los PCW no produce ningún efecto

n = 0	MODE 0	(20 columnas)
n = 1	MODE 1	(40 columnas)
n = 2	MODE 2	(80 columnas)

Esta secuencia de escape produce además (CPC) el borrado de pantalla.

— ESC "A". Sube el cursor a la línea superior.

— ESC "B". Baja el cursor a la línea inferior.

— ESC "C". Mueve el cursor una posición a la derecha.

— ESC "D". Mueve el cursor una posición a la izquierda.

Estas cuatro opciones A, B, C y D no producen ningún efecto, si el cursor se haya en el límite de pantalla correspondiente.

— ESC "E". Borra la pantalla sin afectar a la posición del cursor.

— ESC "H". Sitúa el cursor en la posición o,o de ventana (en los CPC no hay ventanas bajo CP/M, pero en los PCW si). Estas dos últimas secuencias de escape pueden ser utilizadas simultáneamente para producir el borrado y cursor al origen, efecto similar el de la orden Basic CLS de los CPC, en el PCW puede emularse mediante

```
CLS$ = CHR$(27) + "E" + CHR$(27) + "H"
```

Ahora puede utilizarse la variable CLS$ para conseguir dicho efecto simultáneo, mediante

```
PRINT CLS$
```

— ESC "I". Igual que ESC "B" pero produce el scroll de pantalla si el cursor se encuentra en la última línea.

— ESC "J". Borra desde la posición actual del cursor hasta el final de la ventana, sin mover el cursor.

— ESC "K". Borra desde la posición actual del cursor hasta el final de la línea.

— ESC "L". Inserta una línea en blanco desplazando todas las inferiores hacia abajo, sin modificar la posición del cursor.

— ESC "M". Borra la línea en curso subiendo las inferiores un lugar e insertando una en blanco al final de la ventana. No modifica el cursor.

— ESC "N". Borra el carácter situado en la posición del cursor, desplazando el resto de los caracteres de la línea un lugar a la izquierda.

— ESC "X" f c nf nc. Define la ventana de texto, f y c son las coordenadas de origen de la ventana (f = fila, c = columna), nf define el número de filas y nc el de columnas. Dado que f y c están afectados por un parámetro de valor 32 y nf y nc con uno de valor 31 (-1 + 32) la forma adecuada y cómoda de uso es definir una función en la forma.

```
DEF FN FIN$ (f,c,nf,nc) = CHR$(27) + "X" + CHR$(32 + f)
+ CHR$(32 + c) + CHR$(31 + nf) + CHR$(31 + nc)
```

De esta forma la determinación de una pantalla de texto de 32 filas por 90 columnas (max. en los PCW) se haría en la forma

```
PRINT FN FIN$ (0,0,32,90)
```

La definición de ventanas es válida únicamente en los PCW bajo CP/M.

— ESC "Y" f c. Sitúa el cursor en la posición f, c (f = fila, c = columna) indicada. Esta secuencia es equivalente a la orden LOCATE de los CPC y por tanto puede ser emulada en Mallard Basic mediante

```
DEF FN LOCATE$ (f,c) = ESC$ "Y" + CHR$(32 + f) + CHR$(32 + c)
```

y ser utilizada mediante FN en la forma

PRINT FN LOCATE$(f,n) siendo f y c los valores de fila y columna deseados. Detrás de esta sentencia debe incluirse el texto a imprimir en esa posición separado por punto y coma o bien dejar el punto y coma como final de sentencia con el fin de conservar el cursor en esa posición, por ejemplo

```
PRINT FN LOCATE$ (4,5); "TEXTO"
```

o bien

```
PRINT FN LOCATE$ (4,5);
PRINT "TEXTO"
```

— ESC "b" c. Establece el color de la pluma

CPC: Sólo se consideran los 6 bits de la derecha. Estos deben tomarse por parejas para formar los colores básicos.

bits	color base
0 y 1	Azul
2 y 3	Rojo
4 y 5	verde

Los valores de intensidad son tres.

Valor de la pareja de bits	Intensidad
00	nivel 0
01	nivel 1
10	nivel 2

Las diferentes intensidades determinan la gama de colores de los CPC.

PCW: Dado que la pantalla es monocromática, sólo actuan dos valores

Bin	Dec	Color
00111111	63	Claro
00000000	0	Oscuro

— ESC "c" c. Establece el color de la pluma de forma idéntica a lo expuesto anteriormente.

— ESC "d". Borra la ventana desde el principio hasta la posición del cursor.

— ESC "e" y ESC "f". Activan y desactivan el cursor respectivamente.

— ESC "j". Memoriza la posición del cursor, dicha posición puede ser recuperada con ESC "k".

— ESC "o". Borra la línea en curso desde el principio hasta la posición del cursor.

— ESC "p" y ESC "q". Activa y desactiva respectivamente el video inverso (cambio de colores pluma/papel)

— ESC "r" y ESC "u". Activa y desactiva respectivamente el modo subrayado. Unicamente válido para los PCW.

— ESC "v" y "w". Activa y desactiva respectivamente la continuidad de líneas (cuando los caracteres de una línea sobrepasan su longitud, estos aparecen en la siguiente).

— ESC "x". Activa el modo 24 × 80 caracteres que es el que corresponde al MODO 2 de los CPC. Borra simultáneamente la pantalla.

— ESC "y". Desactiva el modo de pantalla anterior.

Apéndice 7. Editor Sencillo para Ficheros de Ordenes

En distintos apartados se ha mencionado la posibilidad de utilizar editores de texto comerciales o bien los facilitados por el sistema operativo CP/M (ED. y RPED), no obstante para aquellos lectores que no dispongan de dichos editores y que el manejo de los CP/M les ofrezca dificultades, se incluye a continuación un pequeño programa en BASIC, que permite su utilización como sencillo editor. Las posibilidades de este editor son mínimas, pero suficientes para crear de forma rápida y sencilla pequeños ficheros, su mayor utilidad es sin duda la creación de ficheros de órdenes y definición del teclado, como los expuestos en los capítulos 0 y 1, de instalación del Mallard Basic.

```
10 ' EDITOR DE TEXTOS SENCILLO
20 ' ........ EDITOR .........
30 CLS$=CHR$(27)+"E"+CHR$(27)+"H"
40 PRINT CLS$
50 INPUT "Nombre del fichero a crear ";NF$
60 INPUT "Maxima longitud de cada linea ";LON
70 PRINT CLS$
80 OPEN "R",1,NF$
90 FIELD 1,LON AS L$
100 PRINT"I N T R O D U C I R     L I N E A S     D E L     F I C H E R O"
105 PRINT:PRINT"Linea                    T e x t o"
110 PRINT STRING$(63,"-")
120 WHILE T$<>"*"
130 K=K+1
140 PRINT K;". ";:LINE INPUT "",T$
150 IF LEN(T$)>LON THEN PRINT"LINEA DEMASIADO LARGA":GOTO 140
160 LSET L$=T$
170 PUT 1,K
180 WEND
190 PRINT "GRABACION Y CIERRE DEL FICHERO *** ";NF$;" ***"
200 CLOSE
210 END
```

OTROS LIBROS PUBLICADOS POR RA-MA

MANUAL PARA TURBO PASCAL; Guía del Programador, J.A. Díaz y L. Suja.

ISBN: 84-86381-27-4. Rústica. 254 páginas.

Este libro es un valioso manual de referencia y consulta para programadores.

Se ha tratado de forma especial el enfoque de cada capítulo, logrando que cada explicación se acompañe de ejemplos, sugerencias, recomendaciones prácticas, detalles de comandos, opciones de las versiones 2.xx y 3.xx, etc.

dBASE III y dBASE II: Guía del Usuario; G. Burns.

ISBN: 84-86381-24-X. Rústica. 200 páginas.

Si usted es el tipo de persona que no se satisface con conocer la mitad de la historia y cree que puede entender lo que hace su ordenador si alguien se lo explica, entonces este es el libro que necesita.

He aquí algunas de sus importantes facilidades:

- 100% dedicado a aplicaciones de negocios reales.
- Explicaciones claras de lo que es una base de datos, y de como dBASE III simplifica este modelo.
- Ejercicios de teclado simples, pero graduados, tanto para el principiante como para el usuario más experto.
- Tratado completo del nuevo sistema dBASE III, con referencias a su predecesor dBASE II.
- Demostraciones de la conversión de programas del dBASE II al dBASE III.
- Explicación de los programas de formateo de pantallas de 8 y 16 bits.
- Listado de programas grandes, completos y funcionando.

CP/M, Guía del Programador. CP/M Plus, 2.2 y 1.4; A. Clarke, J.M. Eaton, D. Powys-Lybbe.

ISBN: 84-86381-18-4. Rústica. 350 páginas.

El libro muestra de forma clara y concisa, como utilizar provechosamente CP/M y CP/M Plus, con ejemplos detallados de comandos y opciones de las principales versiones de CP/M.

Igualmente incluye herramientas de software (lenguajes y editores), junto con una lista que sirve de guía de referencia rápida. Se utilizan ejemplos siempre que se consideran necesarios. Los últimos capítulos le ayudarán a profundizar en CP/M, mostrándole cómo puede organizar y manipular la memoria y el almacenamiento de ficheros.

Estos son algunos de nuestros libros, si desea recibir lista completa solicítela por escrito, o llamando a nuestros teléfonos (91) 200 97 46/47.

Los Ficheros en el AMSTRAD PCW 8256/8512 y MALLARD BASIC para CPCs.

Este libro le permitirá trabajar con MALLARD BASIC, tanto en los modelos AMSTRAD PCW, como en el CPC-6128.

Su cobertura incluye:

- Instalación MALLARD BASIC.
- Manejo de matrices o variables indexadas.
- Estructuras de almacenamiento transitorio.
- Manipulación y ordenación de datos.
- Ficheros: JETSAM, Secuenciales, de Acceso Directo...

Todo su contenido se ilustra con ejemplos y programas de utilidad; y en los lugares apropiados, detalles para su correcta aplicación.

El autor:

Este es el segundo libro de VICTOR J. CAMPO publicado por RA-MA.

El primero: "Los Ficheros en el Amstrad CPC 464/664/6128" contiene la descripción de toda clase de ficheros aplicables a estos, o a cualquier otro modelo de ordenador.

ISBN: 84-86381-30-4

Los Ficheros en el AMSTRAD PCW 8256/8512.

VICTOR J. CAMPO

ra-ma

AMSTRAD CPC

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA

https://acpc.me/

[FRA] Ce document a été préservé numériquement à des fins éducatives et d'études, et non commerciales.
[ENG] This document has been digitally preserved for educational and study purposes, not for commercial purposes.
[ESP] Este documento se ha conservado digitalmente con fines educativos y de estudio, no con fines comerciales.